ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика оцінювання безпеки вебзастосунків із використанням машинного навчання»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Андрій КОНОПЛЯСТИЙ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-61
Андрій КОНОПЛЯСТИЙ

Керівник: Оксана ЗОЛОТУХІНА
канд.техн.наук., доц.

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коноплястому Андрію Руслановичу

1. Тема кваліфікаційної роботи: «Методика оцінювання безпеки вебзастосунків із використанням машинного навчання»

керівник кваліфікаційної роботи Оксана ЗОЛОТУХІНА, канд. техн. наук, доц., доцент кафедри ІПЗ,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, веб-логи користувацьких сесій, поведінкові ознаки, методи машинного навчання для виявлення аномалій, критерії точності оцінювання ризику сесій.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз існуючих методів оцінювання безпеки вебзастосунків та їхніх обмежень.

2. Дослідження методів машинного навчання для виявлення аномалій.
3. Розробка математичної моделі оцінювання ризику.
4. Розробка, навчання і валідація моделі ML та аналіз її ефективності.

5. Перелік ілюстративного матеріалу: *презентація*
   1. Мета, об’єкт та предмет дослідження.
   2. Актуальність роботи.
   3. Завдання роботи.
   4. Математична модель оцінювання аномальності користувацьких сесій.
   5. Практичний результат.
   6. Порівняльний аналіз ефективності моделі.
   7. Висновки.
   8. Апробація результатів дослідження.

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	31.10.25 - 05.11.25	
2	Вивчення матеріалів та даних для дослідження сучасних загроз і вразливостей вебзастосунків	03.11.25 - 07.11.25	
3	Дослідження методів і технологій оцінювання безпеки вебсистем та поведінкового аналізу	08.11.25 - 10.11.25	
4	Аналіз можливостей застосування алгоритмів машинного навчання для виявлення аномальної активності	11.11.25 - 12.11.25	
5	Розроблення концептуальної та математичної моделі оцінювання безпеки на основі машинного навчання	09.11.25 - 13.11.25	
6	Реалізація та тестування моделі ML для класифікації ризикових користувацьких сесій	11.11.25 - 17.11.25	
7	Оформлення роботи: вступ, висновки, реферат	18.11.25 - 23.12.25	
8	Розробка демонстраційних матеріалів	15.12.25 - 19.12.25	

Здобувач вищої освіти _____ Андрій КОНОПЛЯСТИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Оксана ЗОЛОТУХІНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 66 стор., 3 табл., 13 рис., 31 джерел.

Мета роботи – підвищення ефективності оцінювання безпеки вебзастосунків на основі використання алгоритмів машинного навчання для виявлення, аналізу та класифікації аномальної активності.

Об'єкт дослідження – процес оцінювання безпеки вебзастосунків.

Предмет дослідження – методика оцінювання безпеки вебзастосунків на основі використання алгоритмів машинного навчання для виявлення, аналізу та класифікації аномальної активності.

У межах дослідження реалізовано функціонал автоматизованого оцінювання безпеки вебзастосунків у аналітичному середовищі Databricks, що включає збір і обробку поведінкових даних користувацьких сесій, формування аналітичних ознак та обчислення інтегрального показника аномальності. Запропонований підхід базується на використанні структурованих аналітичних таблиць і модуля машинного навчання, що забезпечує масштабованість, гнучкість та можливість подальшого розширення в умовах розподіленої обробки даних.

Розроблено та оптимізовано математичну модель формування інтегрального показника аномальності користувацьких сесій, а також реалізовано модель машинного навчання на основі алгоритму Random Forest для автоматизованого визначення рівня ризику. У результаті експериментальних досліджень встановлено, що запропонована методика забезпечує підвищення точності виявлення аномальної активності порівняно з традиційними rule-based та сигнатурними методами, що підтверджує ефективність використання машинного навчання у задачах оцінювання безпеки вебзастосунків.

Проведено експерименти для оцінювання точності класифікації та порівняння запропонованої методики з традиційними підходами, які показали доцільність застосування поведінкового аналізу та ML-моделей для виявлення прихованих загроз у вебзастосунках.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНКІВ, АНОМАЛЬНА АКТИВНІСТЬ, ПОВЕДІНКОВИЙ АНАЛІЗ, МАШИННЕ НАВЧАННЯ, ОЦІНЮВАННЯ РИЗИКУ, КЛАСИФІКАЦІЯ.

ABSTRACT

Text part of the master's qualification work: 66 pages, 13 pictures, 3 table, 31 sources.

The purpose of the work is to improve the effectiveness of web application security assessment by applying machine learning algorithms for detecting, analyzing, and classifying anomalous activity.

Object of research – the process of evaluating the security of web applications.

Subject of research – a methodology for assessing the security of web applications based on the use of machine learning algorithms for detecting, analyzing, and classifying anomalous activity.

Within the scope of the research, an automated web application security assessment workflow was implemented in the Databricks analytical environment, including the collection and processing of behavioral data from user sessions, the formation of analytical features, and the calculation of an integral anomaly score. The proposed approach is based on the use of structured analytical tables and a machine learning module, which ensures scalability, flexibility and the possibility of further extension under distributed data processing conditions.

A mathematical model for forming an integral anomaly score of user sessions was developed and optimized, and a machine learning model based on the Random Forest algorithm was implemented for automated risk level determination. Experimental results demonstrate that the proposed methodology improves the accuracy of anomalous activity detection compared to traditional rule-based and signature-based methods, confirming the effectiveness of machine learning approaches in web application security assessment.

Experimental studies were conducted to evaluate classification accuracy and to compare the proposed methodology with traditional approaches. The results confirm the feasibility and effectiveness of applying behavioral analysis and machine learning models for detecting hidden threats in web applications.

KEYWORDS: WEB APPLICATIONS, ANOMALOUS ACTIVITY, BEHAVIORAL ANALYSIS, MACHINE LEARNING, RISK ASSESSMENT, CLASSIFICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ML - машинне навчання

AI - штучний інтелект

SAST - статичний аналіз безпеки коду

DAST - динамічний аналіз безпеки

IAST - інтерактивне тестування безпеки застосунків

RASP - захист застосунку під час виконання

WAF - веб-аплікаційний фаєрвол

API - інтерфейс програмування застосунків

HTTP - протокол передачі гіпертексту

HTTPS - захищений протокол передачі гіпертексту

SQL - мова структурованих запитів

XSS - міжсайтовий скриптинг

SQLi - SQL-ін’єкція

RCE - віддалене виконання коду

DoS/DDoS - відмова в обслуговуванні / розподілена відмова в обслуговуванні

CVSS - система оцінювання критичності вразливостей

CSV - формат табличних даних

FPR - частка хибних спрацювань

TPR - частка виявлених атак

ROC - крива залежності чутливості від похибок

JSON - текстовий формат обміну даними

URL - уніфікований локатор ресурсу

OWASP - міжнародний проєкт із безпеки вебзастосунків

ВСТУП

У сучасному цифровому середовищі вебзастосунки відіграють ключову роль у забезпеченні взаємодії між користувачами, сервісами та бізнес-процесами. Через вебінтерфейси здійснюються фінансові транзакції, авторизація користувачів, обробка персональних даних і керування критичними сервісами. Зростання функціональності таких систем супроводжується збільшенням кількості можливих вразливостей, що робить вебзастосунки однією з основних цілей кіберзлочинців [24]. Тому питання забезпечення їхньої безпеки є надзвичайно актуальним як для розробників, так і для організацій, що надають цифрові послуги [23].

Традиційні методи оцінювання безпеки - ручний аудит, сканери вразливостей або сигнатурні системи - здатні виявляти переважно відомі атаки та передбачувані патерни поведінки. Вони недостатньо ефективні у ситуаціях, коли шкідлива активність маскується під звичайні дії користувача, або коли атака здійснюється у новій формі, не відомій системам сигнатурного аналізу. Через це суттєво зростає ризик появи прихованих загроз, що залишаються непоміченими на ранніх етапах [7].

У таких умовах все більшого значення набувають методи машинного навчання, здатні аналізувати поведінкові характеристики користувацьких сесій, виявляти неочевидні закономірності та автоматично класифікувати аномальну активність. ML-моделі дозволяють системі адаптуватися до нових типів атак, зменшувати кількість хибних спрацювань і підвищувати загальну точність аналізу безпеки.

Магістерська робота присвячена розробленню методики оцінювання безпеки вебзастосунків із використанням машинного навчання. Така методика має на меті поєднати можливості поведінкового аналізу, математичного моделювання та алгоритмів класифікації для створення комплексного інструменту виявлення потенційних загроз [10].

Метою роботи - підвищення ефективності оцінювання безпеки вебзастосунків на основі використання алгоритмів машинного навчання для виявлення, аналізу та класифікації аномальної активності.

Об’єктом дослідження - процес оцінювання безпеки вебзастосунків.

Предметом дослідження - методика оцінювання безпеки вебзастосунків на основі використання алгоритмів машинного навчання для виявлення, аналізу та класифікації аномальної активності.

Для досягнення мети роботи було сформовано такі основні завдання дослідження:

1. Проаналізувати сучасні загрози вебзастосунків та існуючі підходи до оцінювання їх безпеки.
2. Дослідити можливості застосування алгоритмів машинного навчання для аналізу поведінкових ознак користувацьких сесій.
3. Розробити математичну модель оцінювання ризику, що включає формування інтегрального показника аномальності.
4. Реалізувати алгоритм визначення рівня ризику сесії та побудувати модель машинного навчання для її класифікації.
5. Провести експериментальну перевірку ефективності запропонованої методики та порівняти результати з традиційними підходами.

Практична цінність роботи полягає у створенні універсальної методики, яку можна інтегрувати у процес розроблення та експлуатації вебзастосунків для автоматичного контролю рівня безпеки. Запропонований підхід дозволяє зменшити витрати на аудит, прискорити аналіз, а також підвищити точність виявлення загроз завдяки адаптивним властивостям моделей машинного навчання.

1 АНАЛІЗ СУЧАСНИХ ЗАГРОЗ ТА ПІДХОДІВ ДО ОЦІНЮВАННЯ БЕЗПЕКИ

1.1 Сучасні загрози та вразливості вебзастосунків

У сучасному цифровому середовищі безпека вебзастосунків перетворилася на один із найкритичніших напрямів діяльності компаній. Вебплатформи зберігають величезні обсяги конфіденційних даних користувачів, тому будь-яка уразливість у коді, архітектурі чи конфігурації може стати причиною серйозних фінансових та репутаційних втрат [15].

Ключова особливість сучасних загроз полягає в їхній складності та комбінованому характері. Якщо раніше атаки здійснювалися переважно окремими методами, то сьогодні зловмисники поєднують кілька векторів впливу - наприклад, спочатку проводять фішинг, потім використовують отримані дані для SQL-ін’єкції або XSS. Через це класичні методи захисту, що базуються на фіксованих правилах, часто виявляються недостатньо ефективними.

Ще одна тенденція - активне використання відкритих бібліотек і фреймворків. Разом із перевагами швидкої розробки вони приносять і ризики, адже навіть одна вразлива залежність може відкрити шлях до всієї системи. Зростає кількість атак, пов’язаних із помилками в налаштуваннях серверів, некоректним обмеженням доступу чи слабким шифруванням даних.

Організація OWASP періодично публікує рейтинг найпоширеніших уразливостей вебзастосунків. За його даними, у 2025 році лідерами залишаються помилки контролю доступу, криптографічні збої та ін’єкційні атаки. Ці типи вразливостей мають найбільший вплив на цілісність і конфіденційність інформації. Нижче на рисунку 1.1 наведено графічне відображення частоти виявлення різних типів загроз за даними OWASP[9].

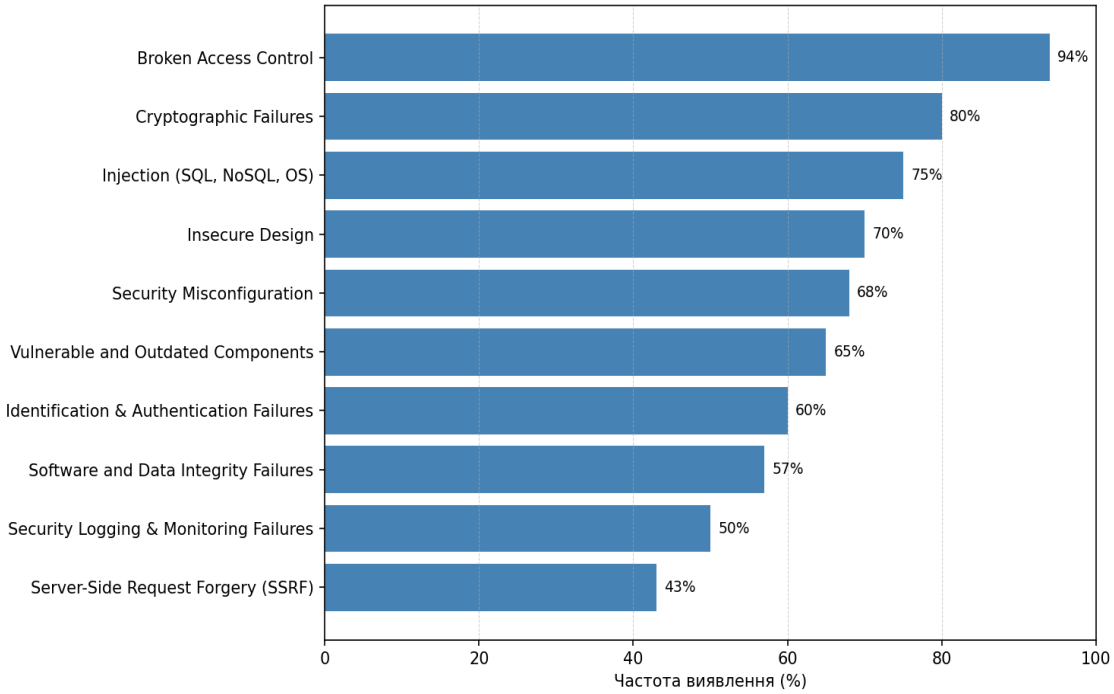


Рис 1.1 Частота основних веб-вразливостей (OWASP Top 10)

1.1.1 Класифікація вразливостей OWASP Top 10

Оцінювання безпеки вебзастосунків неможливе без розуміння найпоширеніших типів уразливостей, які використовують зловмисники. Одним із найавторитетніших джерел у цій сфері є ініціатива OWASP (Open Web Application Security Project), що періодично публікує перелік десяти найкритичніших ризиків безпеки для вебдодатків - OWASP Top 10. Цей список фактично став стандартом галузі, який використовується для оцінювання рівня безпеки під час розробки, тестування та аудиту програмних систем.

До актуальної версії OWASP Top 10 входять такі категорії:

1. Broken Access Control - порушення контролю доступу, яке дозволяє користувачам отримувати права, що їм не належать.
2. Cryptographic Failures - помилки у реалізації або використанні криптографії, через які відбувається витік чутливих даних.
3. Injection - вразливості типу SQL-, NoSQL-, LDAP- або OS-ін’єкцій, що дають змогу виконати шкідливі команди в системі.

4. Insecure Design - архітектурні помилки, коли безпека не була врахована ще на етапі проєктування системи.

5. Security Misconfiguration - некоректні налаштування серверів, бібліотек чи компонентів, які відкривають шлях до експлуатації.

6. Vulnerable and Outdated Components - використання застарілих фреймворків або бібліотек із відомими уразливостями.

7. Identification and Authentication Failures - недоліки у механізмах автентифікації та керування сесіями користувачів.

8. Software and Data Integrity Failures - відсутність перевірки цілісності даних і програмного коду.

9. Security Logging and Monitoring Failures - неналежний моніторинг безпеки, що ускладнює виявлення атак.

10. Server-Side Request Forgery (SSRF) - атаки, коли сервер обманом змушують робити запити до внутрішніх або довірених ресурсів.

Ця класифікація охоплює найкритичніші області, уразливість яких може призвести до втрати конфіденційності, цілісності чи доступності даних. Саме на цих ризиках базуються сучасні системи аналізу безпеки, що використовують як класичні алгоритми перевірки коду, так і моделі машинного навчання. У результаті цього розгляду сформовано розуміння основних напрямів, на які спрямовано виявлення загроз у вебзастосунках.

1.1.2 Типові вектори атак та їх наслідки

Проблематика безпеки вебзастосунків охоплює широкий спектр загроз, що різняться за складністю, ціллю та впливом. У сучасному кіберпросторі зловмисники використовують як класичні, так і нові, більш витончені методи атак, орієнтовані на отримання несанкціонованого доступу, крадіжку даних або порушення роботи сервісу. Важливо розуміти, що більшість таких атак базується на використанні відомих вразливостей, які можна попередити за умови своєчасного виявлення та правильної конфігурації системи[7].

Типовими векторами атак у вебсередовищі є ін’єкційні атаки, міжсайтові скрипти, підміна сесій, експлуатація неправильних налаштувань серверів, атаки типу “відмова в обслуговуванні” (DoS/DDoS), а також спроби використання вразливих або застарілих бібліотек. Кожен із цих векторів має власні технічні особливості та різний рівень впливу на безпеку системи.

Для систематизації даних щодо основних типів атак та їх наслідків наведено таблицю 1.1, у якій узагальнено найпоширеніші приклади та способи їхньої експлуатації.

Таблиця 1.1

Основні типи атак на вебзастосунки та їх характеристики

Тип атаки	Сутність атаки	Потенційні наслідки	Приклади інструментів/методів
SQL Injection	Вставлення шкідливих SQL-запитів через форми введення або URL	Отримання несанкціонованого доступу до бази даних, зміна або видалення інформації	sqlmap, Havij, ручне тестування параметрів запиту
Cross-Site Scripting (XSS)	Вбудовування шкідливого скрипту у сторінку, яку переглядають інші користувачі	Викрадення cookie-файлів, підміна контенту, фішинг	BeEF, XSSer, браузерні експлойти
Cross-Site Request Forgery (CSRF)	Виконання небажаних дій користувачем, який уже авторизований у системі	Підміна запитів від імені користувача, несанкціоновані транзакції	Burp Suite, OWASP ZAP, власні HTML-скрипти
Brute Force / Credential Stuffing	Автоматизований підбір паролів або використання вкрадених облікових даних	Компрометація облікових записів, несанкціонований доступ	Hydra, Metasploit, password lists

Продовження таблиці 1.1

Основні типи атак на вебзастосунки та їх характеристики

Тип атаки	Сутність атаки	Потенційні наслідки	Приклади інструментів/методів
Security Misconfiguration	Помилки в налаштуваннях серверів, фреймворків чи БД	Витік даних, обхід автентифікації, експлуатація відкритих портів	Nmap, Nikto, Shodan
Denial of Service (DoS/DDoS)	Перевантаження сервера великою кількістю запитів	Недоступність вебзастосунку, фінансові збитки	LOIC, HOIC, botnets
Session Hijacking	Перехоплення ідентифікаторів сесії користувача	Керування обліковим записом, підміна даних	Ettercap, Wireshark, браузерні розширення
Directory Traversal	Несанкціонований доступ до файлової системи через URL	Зчитування конфіденційних файлів, компрометація системних даних	Burp Intruder, DirBuster
Outdated Components Exploit	Використання відомих уразливостей у старих версіях бібліотек	Повний контроль над сервером або клієнтом	Exploit-DB, Nessus, OpenVAS
Server-Side Request Forgery (SSRF)	Змусити сервер надсилати запити до внутрішніх ресурсів	Витік внутрішніх даних, обходи мережевих обмежень	SSRFmap, Burp Collaborator

Наведені приклади демонструють, що навіть незначні прорахунки в архітектурі або логіці обробки даних можуть призвести до масштабних наслідків. Наприклад, уразливість XSS часто використовується для викрадення сесійних

токенів, що дає змогу повністю контролювати обліковий запис користувача. Натомість SQL Injection може надати доступ до бази даних, де зберігаються конфіденційні відомості - від персональних даних до платіжної інформації.

Атаки типу Security Misconfiguration і Outdated Components Exploit зазвичай є наслідком недбалості у процесі адміністрування [21]. Їх складно виявити без спеціалізованих інструментів, але саме вони найчастіше стають причиною компрометації корпоративних вебсистем.

Використання систем автоматичного моніторингу безпеки, інтегрованих у CI/CD-процеси, дозволяє своєчасно виявляти подібні вразливості. Проте навіть найсучасніші рішення не гарантують повного захисту без участі людини та постійного аналізу поведінкових аномалій. Саме на цьому етапі доцільно застосовувати машинне навчання, здатне розпізнавати підозрілі дії за непрямими ознаками та кореляціями в трафіку.

У результаті цього аналізу визначено, що комплексне оцінювання безпеки вебзастосунків має ґрунтуватися не лише на сигнатурних методах і скануванні коду, а й на побудові інтелектуальних моделей виявлення загроз. Це відкриває шлях до використання алгоритмів машинного навчання, розглянутих у наступному підрозділі.

1.2 Існуючі методи оцінювання безпеки вебзастосунків

Оцінювання безпеки вебзастосунків включає широкий набір методів - від класичних ручних аудитів до сучасних автоматизованих систем, інтегрованих у життєвий цикл розробки. Кожен підхід має власні сильні й слабкі сторони, тому на практиці доцільно застосовувати комбінацію методів для отримання найбільш повної картини ризиків.

Класифікація основних методів:

1. Ручний аудит і тестування на проникнення. Цей метод передбачає детальний аналіз вихідного коду, архітектури та конфігурацій вебзастосунку спеціалістом із безпеки. Під час тестування використовуються реальні сценарії

атак, що дає змогу виявити складні логічні вразливості, які автоматизовані системи часто пропускають. Ручний аудит залишається найточнішим способом оцінювання, однак потребує значних часових і людських ресурсів.

2. Статичний аналіз коду (SAST). Виконується на етапі розробки й дозволяє знаходити потенційно небезпечні фрагменти у коді до моменту його запуску. Системи SAST інтегруються в CI/CD-процеси й автоматично перевіряють кожен коміт. Це знижує ризики потрапляння критичних вразливостей у продакшн, проте часто генерує надлишкову кількість попереджень, які потребують ручної перевірки.

3. Динамічний аналіз безпеки (DAST). Проводиться вже після розгортання вебзастосунку. Метод моделює поведінку реального користувача, перевіряє точки входу, параметри запитів, форми та API. DAST дозволяє виявляти вразливості, що виникають під час виконання, зокрема пов’язані з некоректною валідацією даних і помилками конфігурації.

4. Інтерактивне тестування (IAST) та RASP. Поєднує принципи SAST і DAST, використовуючи вбудовані агенти, які спостерігають за поведінкою програми під час виконання. IAST надає глибший контекст помилок, а RASP (Runtime Application Self-Protection) забезпечує миттєве блокування атак у реальному часі. Це сучасний напрямок, що активно впроваджується у великих компаніях.

5. Аналіз компонентів і залежностей (SCA). Значна частина вебзастосунків створюється на основі відкритих бібліотек і фреймворків. Метод SCA дозволяє перевірити, чи не містять вони відомих уразливостей, зазначених у базах CVE. Такий підхід особливо ефективний для великих проєктів, де кількість залежностей може перевищувати сотні модулів.

6. Фаззинг (fuzz testing). Це автоматизований процес, під час якого система подає у вебзастосунок випадкові або навмисно некоректні дані, щоб перевірити його реакцію. Фаззинг допомагає знайти критичні помилки, що призводять до збоїв або витоків даних, і добре працює як додатковий рівень захисту до основного тестування.

7. Моделювання загроз (Threat Modeling). Метод передбачає аналітичну роботу з виявлення потенційних сценаріїв атак ще до етапу реалізації. Використовуються моделі типу STRIDE чи DREAD, що допомагають оцінити ризики, пріоритезувати контрзаходи й побудувати архітектуру з урахуванням загроз.

8. Моніторинг і виявлення аномалій. Цей метод охоплює збір логів, аналітику поведінки користувачів та виявлення відхилень у трафіку. На відміну від попередніх підходів, він застосовується вже на етапі експлуатації. У поєднанні з машинним навчанням моніторинг дає змогу автоматично визначати підозрілі дії та реагувати на них до моменту шкоди.

9. Інтеграція тестів безпеки у DevSecOps. Безпека стає невід'ємною частиною життєвого циклу розробки. Автоматизовані сканери, політики якості коду, контроль конфігурацій і ML-аналітика ризиків інтегруються у конвеєри CI/CD, що забезпечує безперервний контроль безпеки та швидке виявлення загроз.

На рисунку 1.2 лінійний графік демонструє узагальнену оцінку основних методів за трьома критеріями: точністю, швидкістю та рівнем автоматизації. На основі зображених тенденцій можна помітити, що ручний аудит забезпечує найвищу точність, проте поступається у швидкості та масштабованості. Автоматизовані методи, такі як SAST, DAST та DevSecOps, навпаки, характеризуються високою продуктивністю та інтеграцією у процес розробки, але потребують додаткової валідації результатів.

Комбіновані рішення, представлені IAST/RASP, демонструють баланс між точністю та автоматизацією, що робить їх оптимальними для корпоративних систем із підвищеними вимогами до безпеки. Аналіз компонентів (SCA) та моніторинг із використанням машинного навчання посилюють ефективність, забезпечуючи постійне відстеження ризиків у реальному часі [26].

Отже, результати порівняльного аналізу підтверджують, що жоден окремий метод не здатен забезпечити повний захист вебзастосунку. Найвищого рівня безпеки можна досягти лише шляхом інтеграції декількох підходів: поєднання статичного, динамічного та інтерактивного аналізу з постійним моніторингом і

машинним навчанням, що дозволяє своєчасно виявляти як технічні, так і поведінкові загрози.

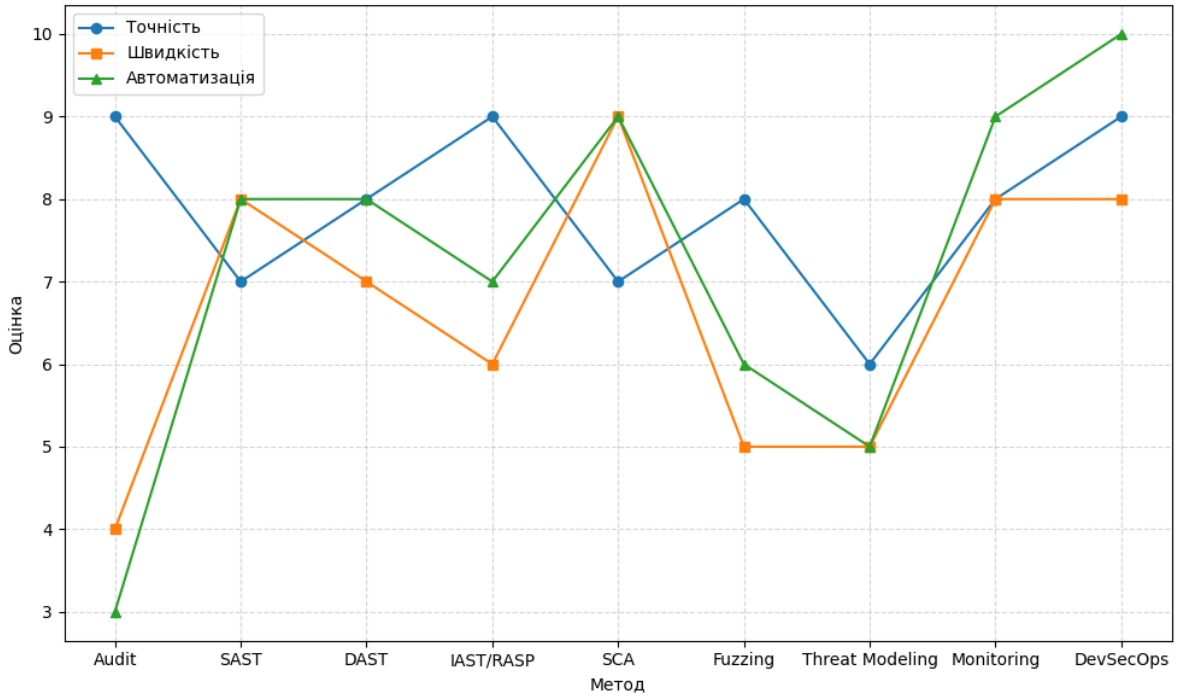


Рис 1.2 Порівняння ефективності методів оцінювання безпеки

1.2.1. Статичний аналіз коду (SAST)

Статичний аналіз коду (SAST, Static Application Security Testing) є одним із базових підходів до оцінювання безпеки вебзастосунків, що передбачає дослідження програмного коду без його виконання. Метою SAST є виявлення потенційних вразливостей у логіці програми, синтаксисі, структурі даних та використанні сторонніх бібліотек на ранніх етапах розроблення [23].

SAST-інструменти аналізують вихідний код, байткод або проміжні подання програми та здійснюють пошук помилок, що можуть призвести до виникнення критичних вразливостей. До таких проблем належать ін’єкційні атаки (SQL/NoSQL/XML Injection), помилки контролю доступу, небезпечні виклики функцій, неправильне оброблення виключень, уразливі шаблони автентифікації та неправильне керування пам’яттю.

Однією з ключових переваг SAST є можливість його застосування на ранніх етапах життєвого циклу розробки програмного забезпечення. Це дає змогу усувати вразливості ще до інтеграції або розгортання системи, що значно знижує витрати на безпеку та скорочує потенційні ризики. Крім того, статичний аналіз забезпечує повне охоплення коду, включно з тими фрагментами, що рідко виконуються або не покриваються тестами.

Втім, SAST має низку обмежень. Цей метод не враховує поведінкові аспекти роботи вебзастосунку, через що не здатний виявити логічні та контекстні вразливості, які проявляються лише під час виконання програми. Також статичний аналіз часто генерує значну кількість хибнопозитивних результатів, що потребує додаткової мануальної перевірки з боку розробників або аналітиків. У разі складних систем із динамічною генерацією контенту SAST може втрачати точність через обмежений доступ до повного виконуваного середовища.

Незважаючи на ці недоліки, SAST залишається важливою складовою комплексного процесу оцінювання безпеки. Його використання рекомендоване у поєднанні з динамічними та поведінковими методами аналізу, що забезпечує більш повне виявлення потенційних загроз та скорочення ризику появи критичних помилок у вебзастосунку.

1.2.2. Динамічний аналіз (DAST)

Динамічний аналіз безпеки вебзастосунків (DAST, Dynamic Application Security Testing) - це підхід, що полягає у тестуванні вебсистеми під час її реального виконання. На відміну від статичного аналізу коду, DAST не потребує доступу до вихідних файлів програми та зосереджується на поведінці застосунку в умовах симуляції взаємодії користувача або зловмисника. DAST-інструменти виконують аналіз через надсилання різноманітних HTTP/HTTPS-запитів, формування тестових сценаріїв взаємодії, моделювання атак і вивчення реакції системи. Такий метод дозволяє виявляти вразливості, що проявляються лише під час виконання вебзастосунку, зокрема:

- некоректну обробку введених даних
- ін’єкційні атаки (SQL/NoSQL Injection, Command Injection)
- помилки в авторизації та контролі доступу
- вразливості у конфігурації вебсервера
- некоректну роботу сесій та cookie
- Cross-Site Scripting (XSS)
- проблеми у бізнес-логіці, що виникають під час взаємодії компонентів системи.

Однією з головних переваг DAST є його здатність працювати з вебзастосунком як із «чорним ящиком», тобто тестувальник не потребує доступу до внутрішнього коду. Це дозволяє оцінювати безпеку системи з позиції зовнішнього користувача або потенційного зловмисника, а також виявляти проблеми, які не фіксуються статичним аналізом.

Попри ефективність у виявленні реальних вразливостей, DAST має низку обмежень. По-перше, він не дозволяє встановити точне місце помилки у коді, що потребує додаткового внутрішнього аналізу. По-друге, покриття коду залежить від сценаріїв тестування: якщо певні частини застосунку не будуть задіяні, відповідні вразливості залишаться непоміченими [9]. По-третє, поведінкові аномалії, що потребують багаторівневого або довготривалого аналізу, DAST виявляє не завжди.

Крім того, DAST не здатний ефективно протидіяти новим або нетиповим атакам, оскільки більшість інструментів використовують бібліотеки відомих вразливостей та правила виявлення, подібні до сигнатурних методів.

Незважаючи на зазначені недоліки, динамічний аналіз є важливою складовою комплексного тестування безпеки. Його застосування у поєднанні зі статичним аналізом, лог-моніторингом та поведінковими методами дозволяє суттєво підвищити рівень захисту вебзастосунків і забезпечити виявлення критичних помилок у різних сценаріях їх роботи.

1.2.3 Комбіновані методи IAST та RASP

Комбіновані методи оцінювання безпеки, зокрема IAST та RASP, поєднують сильні сторони статичного й динамічного аналізу, забезпечуючи більш точне виявлення вразливостей у контексті реальної роботи вебзастосунку. Метод IAST (Interactive Application Security Testing) працює у середовищі виконання та аналізує поведінку програми під час проходження тестових сценаріїв. На відміну від традиційного DAST, який бачить лише зовнішні прояви вразливостей, IAST має доступ до внутрішньої логіки, виконуваних функцій та обробки даних. Це дозволяє точніше визначати місце виникнення потенційних проблем у коді, зменшуючи кількість хибнопозитивних результатів. Такий підхід є ефективним під час модульного або інтеграційного тестування, коли програма працює у контрольованому середовищі і виконуються заздалегідь визначені типові сценарії взаємодії.

RASP (Runtime Application Self-Protection) є більш глибокою технологією, оскільки виконує аналіз безпеки безпосередньо у робочому середовищі вебзастосунку. RASP-модуль інтегрується у застосунок і має можливість контролювати виконання операцій у реальному часі, що дозволяє не лише виявляти підозрілу активність, а й автоматично блокувати її. На відміну від традиційних систем виявлення загроз, RASP розуміє контекст кожної дії в межах виконуваного коду, завдяки чому може визначити, чи є конкретний запит або операція небезпечною саме в поточному стані системи. Такий підхід забезпечує більш адаптивний та оперативний захист, особливо у випадках нових атак, для яких ще не існує сигнатур або заздалегідь сформованих правил.

Попри високу ефективність, IAST та RASP мають і певні обмеження. Для IAST одним із викликів є необхідність тісної інтеграції з тестовим середовищем та залежність від покриття тестами: якщо певні сценарії не виконуються, вразливості, пов’язані з ними, можуть залишитися непоміченими. RASP, у свою чергу, може впливати на продуктивність системи, оскільки постійно стежить за виконанням коду, а також вимагає глибокої інтеграції з архітектурою вебзастосунку.

Лог-аналіз і моніторинг є важливою складовою систем оцінювання безпеки вебзастосунків, оскільки забезпечують постійне відстеження подій, які відбуваються під час роботи сервісу. Файли журналів містять інформацію про запити користувачів, роботу серверних компонентів, спроби автентифікації та інші дії, що можуть свідчити про потенційні загрози. У більшості серверних середовищ лог-файли розміщуються у стандартних директоріях, таких як /var/log, де зберігається велика кількість системних, мережевих та прикладних журналів (див. рис. 1.3).

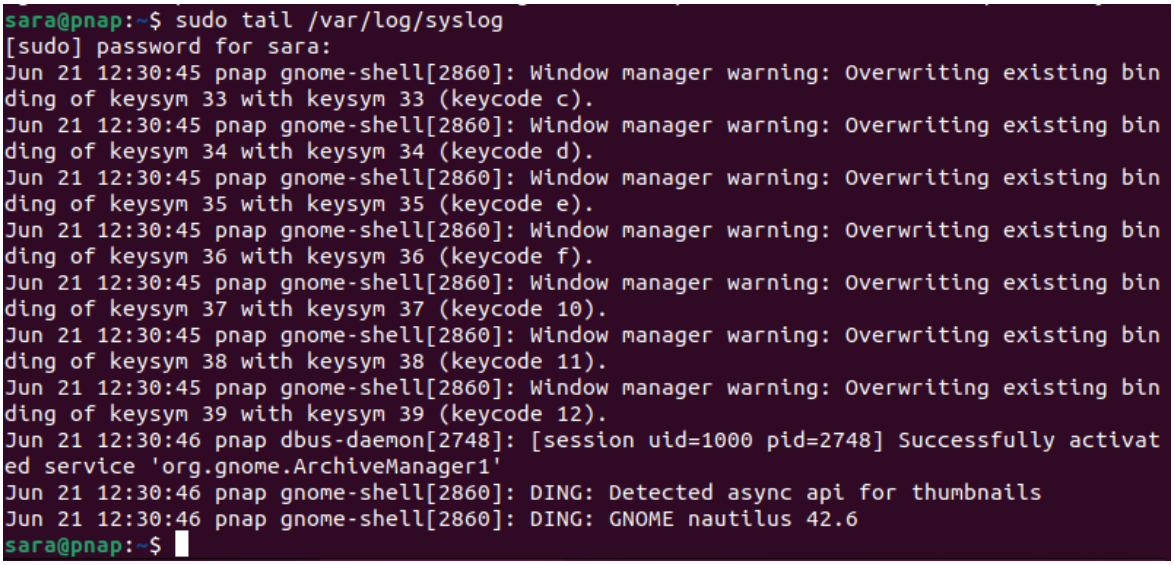


Рис 1.3 Приклад перегляду системного журналу

Під час аналізу логів важливим є не лише фіксація подій, а й виявлення закономірностей у їх появі. Наприклад, часті помилки авторизації можуть свідчити про спроби перебору паролів, а підозріло велика кількість запитів до одного ресурсу про можливу атаку на виснаження. У такому контексті лог-аналіз виступає інструментом раннього попередження, оскільки дозволяє помітити ознаки потенційної атаки задовго до того, як вона призведе до реальних збитків.

Моніторингові системи - такі як Elasticsearch-Stack (ELK), Splunk або Grafana Loki - надають можливість централізовано збирати, агрегувати й інтерпретувати журнали з різних джерел (див. рис. 1.4). Вони автоматично індексують події, виділяють аномальні патерни, формують сповіщення та дозволяють будувати

дашборди для подальшого аналізу. Завдяки цьому фахівці з безпеки можуть виявляти довготривалі поведінкові аномалії, які не відображаються під час одногоразового тестування [15].

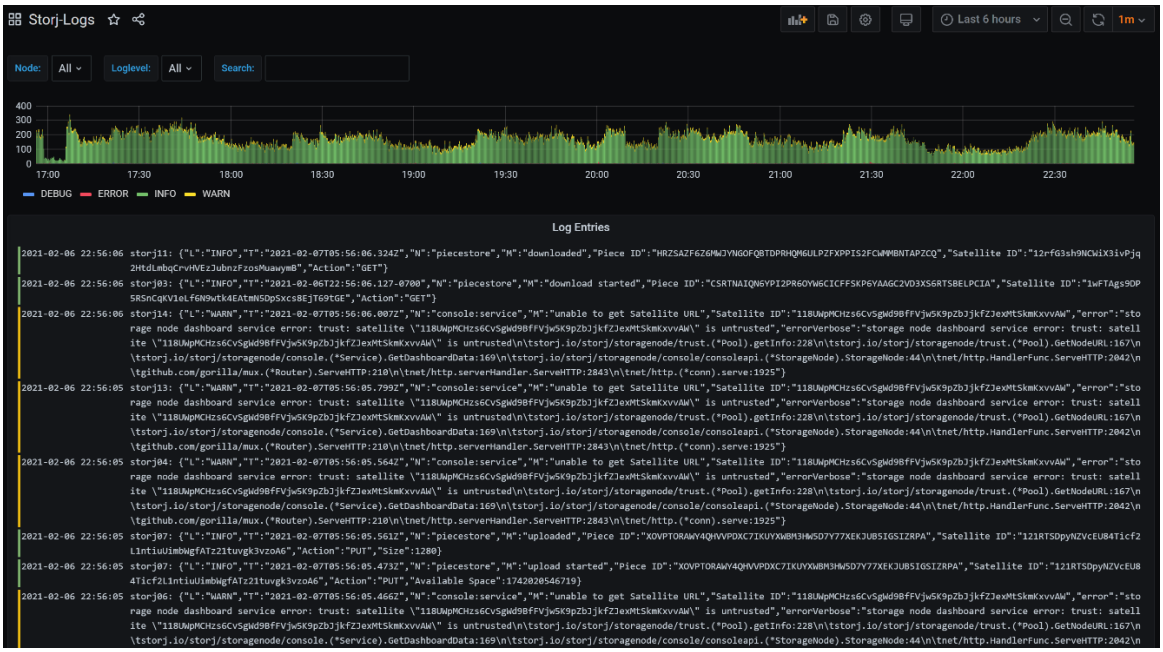


Рис 1.4 Панель Grafana з відображенням логів

Однак традиційний лог-аналіз має обмеження. Він значною мірою залежить від заздалегідь визначених правил, порогів або сигнатур. Це означає, що система здатна реагувати лише на вже відомі типи атак. У випадку нових сценаріїв або слабко виражених аномалій події можуть залишатися непоміченими. Саме тому лог-аналіз часто використовується як основа для побудови більш адаптивних систем на базі машинного навчання, які здатні виявляти нетипову поведінку та аналізувати взаємозв’язки між подіями, що не піддаються простому правилам.

1.3 Проблеми традиційних підходів та необхідність використання ML

Застосування традиційних методів оцінювання безпеки вебзастосунків довгий час залишалося основою практик кіберзахисту. До найпоширеніших підходів належать статичний та динамічний аналіз коду, ручні аудити, сканери вразливостей, сигнатурні системи, а також лог-аналіз і моніторинг у режимі

реального часу. Хоча ці інструменти забезпечують базовий рівень захисту, їхня ефективність істотно знижується в умовах сучасної динамічної загрозової екосистеми. Зростання кількості атак, розширення функціональності вебзастосунків та постійна зміна поведінкових патернів користувачів призвели до того, що традиційні засоби дедалі рідше здатні вчасно та точно виявляти потенційні ризики.

Одним із ключових недоліків класичних методів є їхня орієнтація на відомі та заздалегідь описані вразливості. Технології на кшталт SAST та DAST працюють зі статичними правилами чи наперед визначеними наборами тестів, що дозволяє виявляти лише ті проблеми, які вже зафіксовані в сигнатурних базах або можуть бути відтворені у тестовому середовищі. У випадку появи нових або модифікованих типів атак такі рішення часто виявляються неефективними, оскільки не здатні адаптуватися до невідомих або слабковиражених патернів поведінки.

Другою критичною проблемою є нездатність традиційних інструментів аналізувати контекст взаємодії користувача із системою. У середовищах, де значна частина атак здійснюється легітимними обліковими записами або маскується під нормальну активність, простий пошук сигнатур чи аналіз окремих запитів не дозволяє сформувати цілісне уявлення про поведінкову модель користувача. Наприклад, багатоетапні атаки, які включають поступові зміни активності, адаптивні методи обходу авторизаційних механізмів чи використання поєднаних векторів впливу, часто залишаються непоміченими традиційними засобами.

Лог-аналіз та моніторинг також мають свої обмеження. Попри здатність фіксувати великий обсяг подій, такі системи переважно працюють із порогами та правилами, які потребують ручної настройки. Це призводить до двох критичних наслідків: по-перше, зростає кількість хибнопозитивних сигналів, які перевантажують фахівців із безпеки; по-друге, нетипові або малопомітні аномалії залишаються поза увагою через відсутність відповідного правила. У великих інформаційних системах, де обсяг логів обчислюється мільйонами подій щодня, ручний огляд або класичний rule-based підхід стають практично неможливими [17].

Ще однією суттєвою вадою традиційних методів є низька адаптивність. Системи, побудовані на сигнатурах або жорстко прописаних правилах, не здатні автоматично вдосконалювати свою поведінку у відповідь на нові дані. Коли патерни активності користувачів змінюються, а атаки стають більш витонченими, такі системи починають давати все менше користі. Це особливо помітно в середовищах, де зловмисники використовують «повільні атаки», перепризначені сеанси, бот-мережі або низькоінтенсивні аномалії, що розгортаються у часі.

У підсумку традиційні системи оцінювання безпеки страждають від трьох ключових обмежень:

1. Нездатність виявляти нові атаки через залежність від сигнатур і жорстких правил.
2. Відсутність поведінкового аналізу, що робить неможливим розпізнавання складних аномалій.
3. Високий рівень хибнопозитивних результатів, який збільшує навантаження на фахівців із безпеки та затримує реакцію на реальні інциденти.

Усе це створює реальну потребу у використанні підходів, здатних працювати з великими даними, враховувати контекст подій і виявляти закономірності, що не піддаються простому логічному формалізуванню. На цьому етапі вирішальну роль відіграє машинне навчання. ML-моделі навчаються на історичних даних, аналізують багатовимірні поведінкові ознаки та здатні помічати нетипові відхилення у тих випадках, коли класичні методи демонструють низьку точність. На відміну від сигнатурних засобів, ML-підходи не залежать від попереднього опису атаки: система самостійно визначає статистично значущі патерни й реагує на невідповідності [8].

Перевагою методів машинного навчання є також їхня адаптивність. Моделі можуть автоматично оновлюватися з появою нових даних, що робить їх ефективними навіть у тих випадках, коли характер поведінки користувачів або застосунку поступово змінюється. Крім того, багато сучасних алгоритмів підтримують механізми пояснюваності (explainable AI), що дозволяє фахівцям

інтерпретувати рішення моделі та визначати причини класифікації певних сесій як ризикових.

Таблиця 1.2

Порівняння rule-based та ML-підходів до виявлення загроз

Критерій	Традиційні методи	Підходи на основі ML
Принцип роботи	Фіксовані правила, сигнатури, статичні тести	Навчання на історичних даних, пошук закономірностей
Виявлення нових атак	Низька ефективність, залежність від оновлень	Висока здатність виявляти нові та модифіковані атаки
Поведінковий аналіз	Обмежений або відсутній	Аналіз повних поведінкових патернів користувачів
Адаптивність	Не змінюється без ручного втручання	Автоматичне вдосконалення моделей при появі нових даних
Хибнопозитивні спрацювання	Висока кількість	Значно знижені завдяки багатовимірному аналізу
Швидкість реагування	Залежить від налаштувань та ручної обробки	Реальний час, автоматичне виявлення аномалій
Скалованість	Потребує значних ресурсів при великих об'ємах даних	Легко масштабується для обробки великих потоків подій

Порівняльний аналіз, представлений у таблиці, демонструє ключові відмінності між традиційними підходами до оцінювання безпеки та методами, що базуються на машинному навчанні. Класичні інструменти залишаються ефективними лише в умовах стабільних, відомих загроз і чітко структурованих сценаріїв, тоді як системи на основі ML здатні працювати в умовах мінливої поведінки користувачів і динамічного розвитку атак. Саме багатовимірний аналіз даних, можливість автоматичного вдосконалення моделей та виявлення нових закономірностей забезпечують машинному навчанню істотні переваги у виявленні ризиків. У той час як традиційні методи здебільшого реагують на вже відомі загрози, ML-підходи дозволяють виявляти аномальні дії на ранніх етапах і формувати більш точну та адаптивну систему оцінювання безпеки вебзастосунку.

2 ТЕХНОЛОГІЇ ТА АЛГОРИТМИ МАШИННОГО НАВЧАННЯ У ЗАДАЧАХ БЕЗПЕКИ ВЕБЗАСТОСУНКІВ

2.1 Можливості машинного навчання у задачах кібербезпеки

Машинне навчання відіграє ключову роль у розвитку сучасних систем кіберзахисту, оскільки забезпечує здатність аналізувати великі обсяги даних, виявляти складні закономірності та реагувати на загрози, які неможливо ідентифікувати традиційними методами. На відміну від класичних підходів оцінювання безпеки, що переважно базуються на сигнатурах, фіксованих правилах або статичних тестах, технології машинного навчання дозволяють виявляти як відомі, так і невідомі типи атак, використовуючи поведінкові ознаки та багатовимірний аналіз даних [3].

Однією з ключових можливостей машинного навчання є здатність працювати з даними, які генеруються у процесі взаємодії користувачів із вебзастосунком. Системи авторизації, журнали подій, HTTP-запити, параметри сесій, мережеві характеристики та інші метадані створюють інформаційний простір, у якому можна виявити як нормальні, так і аномальні патерни поведінки. Алгоритми класифікації, кластеризації та аналізу аномалій здатні виділяти такі патерни навіть тоді, коли ознаки атаки не мають чіткої сигнатури або повторюваної форми [11]. Це робить ML особливо ефективним у протидії сучасним атакам, що часто виконуються через серії незначних дій, замаскованих під легітимну користувацьку активність.

Важливою перевагою машинного навчання є можливість побудови поведінкових профілів користувачів. На основі історичних даних моделі можуть визначити, які дії для конкретного користувача або групи користувачів є типовими, а які - потенційно шкідливими. Такий підхід є критичним у виявленні атак на бізнес-логіку, автоматизованих ботів, credential stuffing, а також низькоінтенсивних атак, що відбуваються поступово і не викликають миттєвих підозр у традиційних

систем [20]. Поведінковий аналіз дозволяє виявляти не лише окремі підозрілі запити, а й послідовності дій, що у сукупності визначають ризикову активність.

Ще однією сильною стороною ML у задачах веббезпеки є здатність працювати з багатовимірними ознаками. Типові атрибути, такі як частота HTTP-запитів, довжина параметрів, структура URL, зміна геолокації користувача, характеристики пристрою, часові інтервали між діями та інші поведінкові особливості, можуть бути об’єднані у єдиний вектор ознак. Це дає змогу моделі аналізувати не окремі фактори, а їх взаємозалежності, що значно підвищує точність виявлення аномалій [21].

У практиці захисту вебзастосунків машинне навчання застосовується для широкого спектра задач:

- виявлення атак XSS, CSRF, SQL Injection та SSRF на основі аналізу параметрів запитів;
- ідентифікації шкідливих ботів та автоматизованих скриптів за поведінковими ознаками;
- оцінювання ризику сесій на основі інтегральних показників аномальності;
- прогнозування можливих порушень безпеки на основі історичних даних;
- аналізу структури та змісту вебсторінок для виявлення фішингових кампаній;

Особливе місце займають моделі виявлення аномалій, які не вимагають попередньо позначених даних. У випадках, коли обсяг атак незначний порівняно з нормальними даними, або коли нові типи атак з’являються раніше, ніж їх можна описати, методи аномалійного аналізу дозволяють знаходити відхилення без необхідності визначення чітких правил. Це робить їх незамінними в умовах постійно змінюваного загрозового середовища.

Ще однією суттєвою перевагою ML-систем є адаптивність. На відміну від rule-based механізмів, які потребують ручного оновлення, моделі машинного

навчання здатні автоматично покращувати свої прогнози з появою нових даних. У динамічному середовищі вебзастосунку, де поведінкові патерни користувачів можуть змінюватися залежно від оновлення інтерфейсу, сезонності або оприлюднення нових функцій, адаптивність моделі забезпечує високий рівень точності навіть у довгостроковій перспективі [16].

Сучасні ML-системи також підтримують технології пояснюваності (Explainable AI), які дозволяють інтерпретувати рішення моделі та визначати, які ознаки найбільше вплинули на класифікацію конкретної активності як ризикової. Це є критично важливим для систем безпеки, де кожне рішення має бути поясненим та зрозумілим для аналітиків. Пояснюваність підвищує довіру до автоматизованих систем і полегшує процес валідації результатів.

Окремо варто відзначити здатність ML виявляти складні, багатокрокові атаки, які неможливо ідентифікувати за допомогою традиційних методів. Наприклад, багато сучасних атак складаються з послідовності дій, що окремо є легітимними, але в сукупності формують шкідливий ланцюг. Технології машинного навчання дозволяють аналізувати таку послідовність як єдину поведінкову траєкторію, визначаючи ризики ще до того, як атака буде завершена.

Таким чином, машинне навчання створює передумови для побудови адаптивної, високоточної та інтелектуальної системи оцінювання безпеки вебзастосунків. Його здатність до аналізу поведінкових ознак, виявлення закономірностей, автоматичного оновлення моделей та інтерпретації результатів робить його фундаментальним елементом сучасного підходу до кіберзахисту. У поєднанні з традиційними методами ML утворює комплексний гібридний підхід, що забезпечує значно вищий рівень виявлення загроз та зменшує кількість хибнопозитивних результатів, створюючи надійну основу для оцінювання безпеки вебзастосунків у сучасному цифровому середовищі.

2.1.1. Огляд моделей класифікації та прогнозування ризиків

У задачах оцінювання безпеки вебзастосунків моделі класифікації є одним із ключових інструментів, оскільки вони дозволяють автоматично визначити, чи є

активність користувача нормальною або потенційно шкідливою. На відміну від сигнатурних та rule-based методів, що покладаються на жорсткі правила, моделі машинного навчання виявляють закономірності у поведінці, аналізують взаємозв’язки між ознаками та здатні розпізнавати аномалії навіть тоді, коли ознаки атаки не мають усталеного формату [5]. Це робить їх ефективними у виявленні як класичних загроз, так і нових, адаптивних форм атак.

У практиці кіберзахисту найчастіше використовуються три основні групи моделей: деревоподібні алгоритми, лінійні класифікатори та глибинні нейронні мережі. Деревоподібні алгоритми, зокрема Decision Tree, Random Forest і Gradient Boosting, демонструють високу точність на різнорідних даних та забезпечують інтерпретованість результатів, що є критичним аспектом у системах безпеки. Лінійні моделі, такі як логістична регресія або SVM, характеризуються високою швидкістю класифікації та добре підходять для оброблення великої кількості простих запитів у реальному часі. Глибинні нейронні мережі застосовуються для аналізу складних поведінкових структур, послідовностей HTTP-запитів та багатокрокових атак, які важко розпізнати традиційними методами [29].

Щоб систематизувати ключові відмінності між моделями, наведено узагальнену таблицю 2.1, у якій порівнюються їхні переваги, недоліки та типові сфери застосування у контексті кібербезпеки вебзастосунків.

Таблиця 2.1

Порівняння моделей машинного навчання для виявлення аномалій

Модель	Переваги	Недоліки	Типові сценарії використання
Decision Tree	Інтерпретованість, швидке навчання	Перенавчання	Базові системи виявлення аномалій
Random Forest	Висока точність, стійкість до шуму	Повільна інференція при великій кількості дерев	Виявлення XSS, SQLi, CSRF

Продовження таблиці 2.1

Порівняння моделей машинного навчання для виявлення аномалій

Модель	Переваги	Недоліки	Типові сценарії використання
Gradient Boosting (XGBoost)	Гнучкість, висока продуктивність	Потребує тонкого налаштування	Аналіз складних поведінкових атак
Logistic Regression	Висока швидкість, низькі витрати ресурсів	Погана робота з нелінійними залежностями	Фільтрація простих загроз
SVM	Точність на малих наборах	Погана масштабованість	Аналіз параметрів запитів
LSTM	Аналіз часових послідовностей	Висока потреба у даних	Виявлення «повільних» багатокрокових атак
CNN	Автоматичне виділення ознак	Висока вартість навчання	Аналіз структури HTTP-трафіку

Порівняння моделей демонструє, що жоден алгоритм не є універсальним. Деревоподібні ансамблі забезпечують баланс між точністю та інтерпретованістю і є ефективними для аналізу великих наборів поведінкових ознак. Лінійні класифікатори доцільно застосовувати у високонавантажених системах, де критично важлива швидкість реагування. Нейронні мережі найкраще працюють у тих випадках, коли необхідно аналізувати часові залежності або складні структури трафіку, що характерно для сучасних атак на бізнес-логіку та послідовні багатоетапні вторгнення.

2.1.2. Поведінкові ознаки в аналізі користувацьких сесій

Ефективність моделей машинного навчання у задачах виявлення аномалій значною мірою залежить від якості сформованих ознак, які описують поведінку користувача у межах певної сесії. Поведінкові ознаки (behavioural features) відображають спосіб взаємодії користувача з вебзастосунком, включно з характеристиками HTTP-запитів, параметрами навігації, частотою операцій,

часовими інтервалами та іншими динамічними властивостями. Саме ці ознаки дозволяють моделі відрізняти типову активність від нетипових відхилень, що можуть свідчити про спробу атаки [12].

Поведінковий аналіз базується на тому, що кожен користувач має властивий йому патерн взаємодії: спосіб перегляду сторінок, темп виконання операцій, частоту переходів та рівень відхилення від звичайних сценаріїв [21]. Будь-яка суттєва зміна у цьому патерні може сигналізувати про шкідливу активність, наприклад про автоматизований бот-трафік, скомпрометований акаунт або багатокрокову поведінкову атаку. Згідно з дослідженнями ZeroThreat та Akamai, поведінковий аналіз є одним із найефективніших методів виявлення складних атак, що не мають явної сигнатури.

Однією з ключових переваг поведінкових ознак є їх стійкість до модифікацій з боку зловмисника. Якщо у параметрах запитів можна змінити структуру або форму, то змінити сам тип поведінки - наприклад, темп дій, послідовність переходів або характер взаємодії з UI - значно складніше. Саме тому поведінкові ознаки часто використовуються для виявлення атак на бізнес-логіку, credential stuffing, сканування структур сайту та аномалій, пов’язаних із відхиленням від нормальних користувацьких шаблонів.

Найпоширеніші групи поведінкових ознак включають:

1. Ознаки активності (FA - Frequency Activity). Включають кількість запитів за одиницю часу, частоту оновлення сторінок, кількість POST-операцій або спроб авторизації. Висока активність може сигналізувати про автоматизовані атаки або бот-трафік.

2. Ознаки помилок (ER - Error Rate). Відображають відсоток некоректних запитів, помилок авторизації, повторів неправильної сесії, некоректно сформованих параметрів. Аномально високий показник помилок може свідчити про перебір паролів або спроби обходу валідаційних механізмів.

3. Ознаки сесійної логіки (UFL - User Flow Logic). Характеризують логіку переходів користувача між сторінками. Для типового користувача характерні

звичні шляхи навігації, тоді як для зловмисника - хаотичні або сканувальні переходи.

4. Ознаки швидкості взаємодії (SP - Speed Patterns). Включають часові інтервали між діями. Людина взаємодіє з інтерфейсом нерівномірно, тоді як бот - із надмірною регулярністю.

5. Ознаки аномалій параметрів (A - Anomaly Attributes). Це відхилення у довжині полів, аномальні параметри cookie, незвичні заголовки, змінені User-Agent або нехарактерні токени.

Ці групи ознак у сукупності дозволяють моделі формувати детальний поведінковий профіль користувача, що є основою для виявлення неочевидних атак. Вибір ефективних поведінкових ознак є критично важливим, оскільки моделі машинного навчання працюють не з самими подіями, а з їхніми числовими чи категоріальними представленнями. Дослідження показують, що точність класифікації та рівень виявлення аномалій безпосередньо залежать від правильності вибору ознак та способу їх нормалізації. Саме тому більшість сучасних платформ безпеки використовують комбіновані вектори ознак, які одночасно включають активність, швидкість дій, логіку переходів та аномалії параметрів запитів [29].

2.2 Технології та методи формування вектора ознак для моделі машинного навчання

Формування вектора ознак є ключовим етапом у створенні системи оцінювання безпеки вебзастосунків, оскільки саме від якості цього етапу залежить здатність моделі машинного навчання коректно інтерпретувати поведінку користувачів. Якщо попередній підрозділ зосереджувався на типах поведінкових характеристик, то цей розділ описує технологічний та алгоритмічний процес їх перетворення на структурований набір числових значень, з якими працює ML-модель.

Основна складність формування ознак полягає в тому, що вихідні дані вебсервера є неструктурованими: журнали подій містять хаотичні HTTP-запити, параметри різної довжини, різний формат часових позначок, службові записи та помилки. Тому першим етапом є очищення та нормалізація даних. Сюди входить усунення дублікатів, синхронізація часових позначок, уніфікація форматів заголовків, фільтрація бот-трафіку, що не є предметом аналізу, та вирівнювання некоректно сформованих полів запиту. На цьому рівні формується первинна структурована таблиця подій [16].

Другим критично важливим етапом є ідентифікація та групування сесій. Оскільки окремий HTTP-запит не містить достатньої інформації для аналізу, події агрегують у поведінкові блоки. Сесія може формуватися на основі cookie-параметрів, унікальних токенів, комбінації User-Agent + IP або за принципом часових вікон неактивності. На основі цих сесій надалі обчислюються статистичні характеристики, що відображають поведінку певного користувача.

Після групування даних відбувається агрегація подій у числові характеристики. На цьому етапі сирі дії перетворюються на статистично значимі величини. Наприклад:

- загальна кількість запитів у межах сесії;
- кількість унікальних сторінок;
- медіанний та середній інтервали між запитами;
- кількість помилкових запитів;
- максимальна глибина переходів;
- частка нестандартних параметрів URL;
- частота повторюваних дій;

Ці параметри не є поведінковими ознаками самі по собі, а слугують основою для подальшого побудови вектора ознак (feature vector), тобто фінального набору числових значень, який отримує модель машинного навчання.

Наступним важливим етапом є нормалізація та масштабування ознак. Значення, що суттєво відрізняються за діапазоном (наприклад, довжина параметрів

може бути від 5 до 3000, а кількість переходів - від 1 до 15), повинні бути вирівняні. Застосовуються методи Min-Max Scaling, Standard Scaling або логарифмічні трансформації. Це запобігає домінуванню окремих ознак і дозволяє моделі рівномірно враховувати всі параметри.

Суттєвою частиною етапу є і кодування категоріальних даних. Параметри типу HTTP-методу, типу помилки, типу пристрою, країни, ОС або браузера перетворюються у числові формати за допомогою one-hot encoding, label encoding або binary encoding [6]. Вибір методу залежить від розмірності та частотності категорій. На наступному рівні здійснюється усунення надлишкових ознак. Методами кореляційного аналізу, тестів або алгоритмів визначення важливості (наприклад, Random Forest Feature Importance) виключаються ознаки, які не впливають на рішення моделі або створюють шум. Це підвищує точність класифікації та знижує ризик перенавчання.

У фінальній частині формується остаточний вектор ознак, який представляє кожну сесію у вигляді компактного багатовимірного числового опису. Такий вектор може містити від 20-30 базових параметрів у простих моделях до 200-300 ознак у комплексних системах виявлення атак. На цьому етапі всі показники мають однакову форму, масштаб і формат, що дозволяє передати їх на вхід класифікатору.

2.2.1. Техніки нормалізації та масштабування ознак перед поданням на вхід моделі

Після формування вектора ознак важливим етапом підготовки даних є нормалізація та масштабування параметрів. Оскільки різні ознаки мають різні одиниці вимірювання та діапазони значень, модель машинного навчання може надавати непропорційну вагу тим параметрам, які мають великі числові масштаби. Це знижує точність класифікації та збільшує ризик перенавчання. Тому приведення всіх ознак до узгодженого формату є обов’язковим етапом у побудові системи автоматичного оцінювання безпеки вебзастосунків.

Першим кроком є масштабування кількісних параметрів, таких як кількість HTTP-запитів у сесії (TR), кількість унікальних endpoint-ів (UE) або середня

довжина запиту (AR). У сирих даних ці параметри можуть значно відрізнятися між собою: наприклад, тривалість сесії вимірюється у секундах, кількість запитів - у десятках, а довжина параметрів - у сотнях символів [18]. Якщо передати ці значення без масштабування, модель буде схильна «зважувати» великі числа суттєво вище за дрібні. Для усунення цього ефекту застосовують методи Min–Max Scaling або Standard Scaling.

Min-Max Scaling використовується у випадках, коли модель потребує обмеження ознак у межах [0;1]. Це зручно для інтегральних показників, логічних лічильників та параметрів поведінки, які мають природну верхню межу. Наприклад, показник частоти помилок авторизації або частка некоректних запитів з IP (IER) зручно нормуються за цим принципом.

Standard Scaling (z-нормалізація) застосовується у випадках, коли розподіл значень є нерівномірним або містить викиди. Метод дозволяє центровано змістити дані навколо середнього значення та привести дисперсію до одиниці. Це особливо корисно для ознак, які залежать від активності користувача та можуть суттєво змінюватися залежно від навантаження на систему, наприклад показник TS (тривалість активності).

Окрему увагу приділяють нормуванню бінарних і категоріальних параметрів. Дані такого типу, включно зі значеннями успішних та неуспішних входів (USL, UFL), не можуть бути подані моделі у первісному вигляді. Їх слід перетворити в числові формати за допомогою методів One-Hot Encoding або Label Encoding. Хоча ансамблеві моделі, такі як Random Forest, менш чутливі до масштабування, структурування ознак зберігає цілісність моделі та мінімізує вплив некоректно підготовлених даних.

Наступним етапом є усунення взаємної кореляції ознак. Деякі параметри можуть бути тісно пов’язані між собою, наприклад TR (total requests) та ITR (requests from IP). Сильна кореляція створює надмірність, яка негативно впливає на узагальнюючу здатність моделі. Кореляційний аналіз або методи відбору ознак

дають змогу виключити дублюючі параметри та залишити лише ті, що суттєво впливають на класифікацію [21].

Фінальним етапом є перевірка стабільності ознак на тестових вибірках. Це означає, що нормування та масштаби значень повинні залишатися незмінними у всіх етапах - від навчання до розгортання моделі. Для цього використовують збережені параметри нормалізації (наприклад, μ і σ у Standard Scaling), які застосовуються до нових сесій під час реальної роботи моделі. Таким чином, нормалізація, масштабування та відбір ознак забезпечують коректну роботу моделі машинного навчання, мінімізують викривлення у даних та підвищують точність виявлення аномальної активності. Для побудови інтегрального показника аномальності (AAS) та подання вектора ознак до класифікатора Random Forest ці процедури є обов'язковими, оскільки гарантують узгодженість даних і відтворюваність результатів.

2.3 Математична модель інтегрального показника аномальності

У сучасних системах забезпечення безпеки вебзастосунків особливу роль відіграє можливість своєчасного виявлення нетипової або потенційно небезпечної поведінки користувача. На відміну від традиційних підходів, що орієнтуються переважно на сигнатурні ознаки атак або статичні правила, методи поведінкового аналізу дозволяють враховувати широкий спектр параметрів взаємодії користувача із системою. При цьому виникає потреба у створенні універсальної числової метрики, яка б відображала ступінь "аномальності" сесії та могла використовуватися як для автоматичної класифікації, так і для подальших рішень системи безпеки.

Такою метрикою є інтегральний показник аномальності (AAS) - математично сформульована характеристика, яка враховує різні типи поведінкових ознак, зведені до нормованої шкали. AAS дозволяє не лише порівнювати між собою різні

сесії, але й створює можливість адаптації моделі машинного навчання до різних патернів активності, що є критично важливим у динамічному середовищі кіберзагроз.

Побудова AAS має на меті:

1. об’єднання розрізнених поведінкових даних у єдину узагальнену характеристику;
2. усунення впливу різних масштабів ознак на процес класифікації;
3. створення формалізованої основи для моделей машинного навчання;
4. підвищення чутливості системи до малопомітних поведінкових аномалій.

2.3.1. Формування вектора ознак користувацької сесії

Першим етапом побудови математичної моделі є формування вектора ознак, який описує користувацьку активність у вигляді набору числових параметрів. Вектор охоплює як частотні характеристики, так і показники взаємодії з інтерфейсом, інформацію про поведінкові відхилення, а також ознаки, пов’язані з аналізом помилок та IP-активності. Структура вектора має вигляд:

$$X = [\, TR, FA, UE, AR, TS, AAS, ITR, IER, UFL, USL \,] \tag{2.1}$$

де:

TR – загальна кількість $HTTP$ – запитів у сесії;

FA – невдалі спроби автентифікації;

UE – кількість унікальних $endpoint$ – ів;

AR – середня довжина запиту;

TS – тривалість активності;

AAS – інтегральний показник аномальності;

ITR – кількість запитів від IP;

IER – частка помилкових запитів з IP;

UFL – невдалі входи користувача;

USL – успішні входи користувача.

Важливим аспектом є те, що вектор включає різнотипні ознаки, які мають різну природу: від лічильників до характеристик часових інтервалів та похідних поведінкових індикаторів. Саме така різнорідність дозволяє моделі точніше ідентифікувати відхилення.

Однак різнорідність параметрів потребує нормування, оскільки модель не може інтерпретувати ознаки з різною шкалою рівноправно. Тому кожна ознака проходить етап масштабування до інтервалу [0;1], що забезпечує коректне використання у формулі AAS та подальшій класифікації.

2.3.2. Формула інтегрального показника аномальності

Інтегральний показник AAS визначається як зважена сума нормованих поведінкових ознак:

$$AAS = 0.25 \cdot \min\left(\frac{FA}{10}, 1\right) + 0.2 \cdot \min(ER, 1) + 0.2 \cdot \min\left(\frac{UFL}{10}, 1\right) + 0.2 \cdot I(SP) + 0.15 \cdot I(A) \quad (2.2)$$

де:

$\boldsymbol{FA}$ – кількість невдалих спроб входу;

$\boldsymbol{ER}$ – частка помилкових запитів, що надходять з IP – адреси;

$\boldsymbol{UFL}$ – кількість невдалих спроб входу користувача;

$\boldsymbol{SP}$ – бінарний індикатор, що набуває значення 1 у разі виявлення підозрілих поведінкових патернів у подіях сесії та 0 – за їх відсутності

$\boldsymbol{A}$ – кількісний показник аномальних подій у веб – логах, зафіксованих у межах користувацької сесії;

$\boldsymbol{I(x)}$ – одинична функція, що приймає значення 1 у разі наявності події.

У моделі додатково використовується подієвий (rule-based) індикатор, який дає змогу врахувати наявність ключових поведінкових аномалій, що мають

високий пріоритет у контексті безпеки. Кожний із параметрів цього індикатора має окреме семантичне значення та відображає певний тип небезпечної активності.

Зокрема, параметр FA характеризує кількість невдалих спроб входу до облікового запису, що є типовою ознакою brute-force атак, коли зловмисник намагається підібрати пару логін/пароль. Значення ER відображає частку помилкових або некоректно сформованих HTTP-запитів, що надходять з певної IP-адреси, і часто вказує на використання автоматизованих інструментів сканування, fuzzing або некоректної взаємодії бота [13]. Параметр UFL фіксує невдалі логіни користувача та дозволяє виявити спроби експлуатації вразливостей автентифікації або використання викрадених облікових даних. Показник SP відповідає за наявність підозрілих або нетипових поведінкових патернів, які відхиляються від звичайної логіки користувацьких дій і можуть свідчити про автоматизацію або зловмисну взаємодію. Ознака A відображає виявлені аномалії у веб-логах, такі як повторювані помилки, нестандартні структури параметрів чи атипові послідовності дій.

Формування значення одиничної функції I(x) здійснюється на основі попередньо визначених правил та порогових значень для кожної поведінкової ознаки. Для цього на аналітичному рівні даних виконуються перевірки умов, що відповідають типовим сценаріям небезпечної активності, зокрема перевищення допустимої кількості невдалих спроб входу, аномально високої частки помилкових запитів, нетипової послідовності подій або наявності підозрілих шаблонів у веб-логах.

У разі виконання відповідної умови значення функції I(x) для даної події встановлюється рівним 1, у протилежному випадку - 0. Таким чином, процес формалізації подій базується не на евристичному припущенні, а на чітко визначених логічних правилах, що забезпечує прозорість і відтворюваність результатів. Завдяки цьому rule-based індикатор дозволяє сконцентрувати інформацію про критичні події та доповнює математичну модель AAS, виступаючи додатковим елементом поведінкового аналізу.

Для інтерпретації результатів роботи моделі вводиться шкала оцінювання, у межах якої інтегральний показник або ймовірність аномальної поведінки поділяються на три рівні ризику:

0.0-0.3 - нормальна активність;

0.3-0.7 - сесія з ознаками потенційної аномалії;

0.7-1.0 - сесія з високою ймовірністю атаки.

Пояснення логіки інтегрального показника ґрунтується на оцінці впливу кожної нормованої поведінкової ознаки на загальний ризик сесії. Надмірна кількість HTTP-запитів (TR) може бути свідченням бот-активності або масового сканування структури вебзастосунку. Значна кількість невдалих спроб входу (FA, UFL) є характерним маркером brute-force атаки, коли зловмисник намагається підібрати пароль шляхом автоматизованого перебору. Підвищена частка некоректних або помилкових запитів (IER) зазвичай пов’язана зі спробами визначення вразливостей за допомогою fuzzing-інструментів. Якщо користувач звертається до великої кількості унікальних endpoint-ів (UE), це може свідчити про дослідження внутрішньої структури системи або спроби доступу до прихованих чи службових ресурсів. Висока інтенсивність запитів із конкретної IP-адреси (ITR) типова для автоматизованих бот-мереж. Тривалість сесії (TS) також є важливим поведінковим параметром: короткі монотонні сесії характерні для ботів, тоді як надмірно довгі - можуть бути ознакою повільних, багатокрокових атак [30].

У сукупності кожна ознака формує окремий аспект поведінкового профілю користувача, а їх зважене об’єднання в межах математичної моделі AAS дозволяє отримати комплексну оцінку рівня аномальності сесії. Такий підхід забезпечує більш точне та стійке виявлення шкідливої активності, оскільки враховує як кількісні, так і логічні та поведінкові характеристики користувача.

2.3.3. Ймовірнісний показник ризику RiskScore

У процесі оцінювання поведінки користувача інтегральний показник AAS слугує важливим індикатором ступеня аномальності сесії, однак сам по собі він не є класифікаційним результатом. Для визначення того, чи належить сесія до

категорії «нормальна» або «аномальна», використовується ймовірнісний показник RiskScore, який є вихідним значенням моделі машинного навчання. RiskScore відображає оцінку моделі щодо того, наскільки поточна сесія відповідає поведінці, характерній для шкідливих або підозрілих дій.

Математично RiskScore визначається як умовна ймовірність того, що сесія належить до класу аномальної активності за умови заданих поведінкових ознак:

$$RiskScore = P \cdot (label = 1 \; or \; X) \tag{2.3}$$

де:

$\boldsymbol{label = 1}$ – сесія класифікована як аномальна,

$\boldsymbol{label = 0}$ – сесія є нормальною,

$\boldsymbol{X}$ – вектор ознак користувача.

Таким чином, на відміну від інтегрального показника AAS, який є статичною зваженою сумою нормованих параметрів, RiskScore формується на основі нелінійної моделі машинного навчання, що враховує взаємозв'язки між ознаками та приховані закономірності у поведінці користувачів. Модель Random Forest, яка застосовується для класифікації, обчислює RiskScore шляхом агрегування прогнозів окремих дерев прийняття рішень, кожне з яких голосує за певний клас. Остаточне значення RiskScore відповідає частці дерев, що класифікували сесію як аномальну. Такий механізм робить оцінку стійкою до шуму у даних, підвищує загальну точність та знижує ризик перенавчання.

RiskScore виступає універсальною метрикою, яка може використовуватися як для автоматичного прийняття рішень, так і для відображення у системах моніторингу безпеки [11]. Наприклад, при низьких значеннях RiskScore (0.0-0.3) асистема може обмежитися логуванням активності, тоді як для середнього рівня ризику (0.3 - 0.7) можуть застосовуватися превентивні дії, як-от додаткова перевірка користувача або аналіз сесії аналітиком безпеки.

Значення RiskScore, що перевищують поріг 0.7, зазвичай сигналізують про високу ймовірність атаки, що потребує негайного реагування: блокування сесії, обмеження доступу або запуск механізмів багатоетапної автентифікації.

У практичних системах RiskScore нерідко використовується разом з інтегральним показником AAS. Обидва показники виконують різні функції: AAS забезпечує інтерпретованість і прозорість моделі, тоді як RiskScore відображає гнучку, нелінійну оцінку ризику. Поєднання цих двох метрик дозволяє системі безпеки одночасно бути і зрозумілою для експертів, і високоефективною у виявленні складних та малопомітних атак. Таким чином, RiskScore є невід’ємною частиною математичної моделі оцінювання безпеки, забезпечуючи точну ймовірнісну класифікацію користувацьких сесій та створюючи основу для подальших автоматизованих або ручних рішень у системах захисту вебзастосунків.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ МЕТОДУ ОЦІНЮВАННЯ БЕЗПЕКИ ВЕБЗАСТОСУНКІВ

3.1 Архітектура запропонованого методу оцінювання безпеки вебзастосунків

Запропонований метод оцінювання безпеки вебзастосунків побудований як послідовний процес обробки даних, у межах якого журнали подій вебзастосунку перетворюються на вектор ознак користувацьких сесій, далі - на інтегральний показник аномальності та, врешті, на рішення про рівень ризику. Архітектура методу реалізована у вигляді конвеєра, де кожен модуль виконує чітко визначену функцію, а результат його роботи використовується на наступному етапі [14].

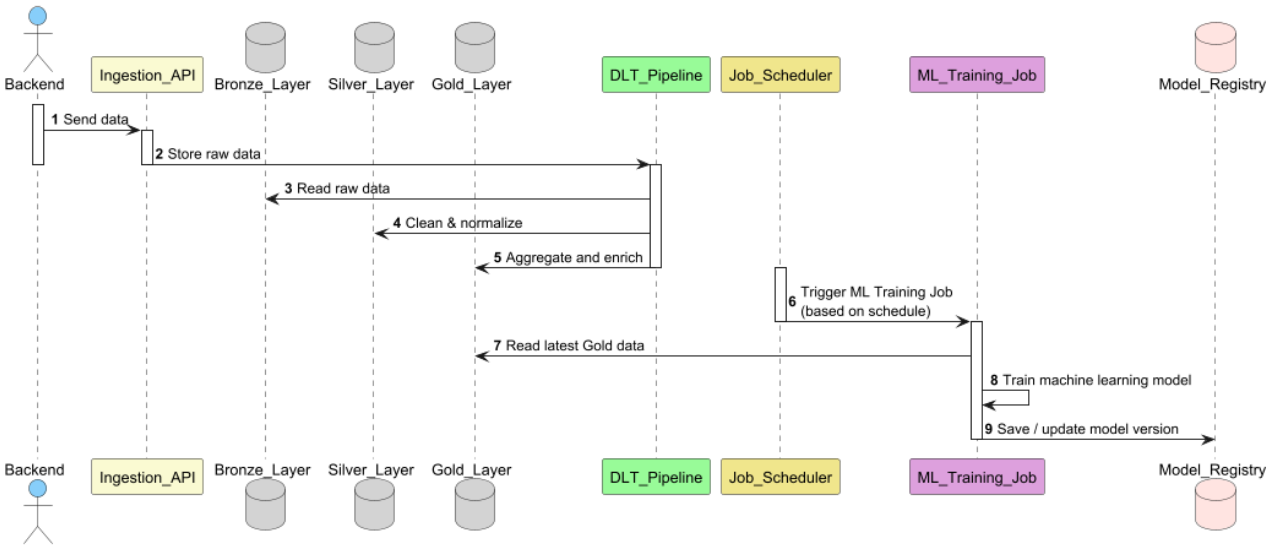


Рис 3.1 Загальна архітектура методу оцінювання безпеки вебзастосунків

На діаграмі 3.1 відображено основні компоненти та обмін даними між ними. Логіка роботи методу може бути описана покроково таким чином:

1. Користувач взаємодіє з вебзастосунком. Під час переходів між сторінками, виконання авторизації, надсилання форм та інших дій формуються HTTP-запити, які обробляються бекендом вебзастосунку.

2. Вебзастосунок генерує журнали подій (логи) та передає їх у модуль збору. Для кожного запиту фіксуються метод, URL, код відповіді, параметри, IP-адреса, ідентифікатор користувача або сесії, часові мітки та інші технічні атрибути.
3. Модуль збору даних зберігає сирі логи в сховищі. На цьому етапі дані ще не агреговані й можуть містити шум, технічні записи або службові запити, але зберігаються у повному обсязі для подальшої обробки.
4. Модуль попередньої обробки читає сирі журнали, очищує та нормалізує їх. Видаляються дублікати, коригуються часові пояси, уніфікуються формати полів, логи групуються за сесіями користувача. У результаті формується структурований набір подій, який відображає послідовність дій у межах кожної сесії.
5. Модуль формування ознак агрегує дані на рівні сесії та обчислює поведінкові параметри. Для кожної сесії розраховуються показники інтенсивності запитів, кількість унікальних endpoint-ів, частка помилкових запитів, тривалість активності, кількість успішних та невдалих входів, індикатори rule-based аномалій тощо. На основі цих значень формується вектор ознак, який описує поведінку користувача в компактному числовому вигляді.
6. На основі сформованих ознак обчислюється інтегральний показник аномальності. Поведінкові параметри, що характеризують сесію, поєднуються у єдину метрику, яка відображає ступінь відхилення від типової активності. Отримане значення використовується як додаткова ознака і як окремий показник ризику для аналітичних цілей.
7. Вектор ознак разом з інтегральним показником передається до моделі машинного навчання. Класифікаційна модель, навчена на історичних даних, оцінює ймовірність того, що поточна сесія належить до класу аномальної або шкідливої активності, й формує ймовірнісний показник RiskScore.

8. Модуль прийняття рішень інтерпретує RiskScore та генерує відповідну дію. Якщо значення ризику низьке, сесія вважається нормальною і додаткових дій не виконується. Для середніх значень можуть застосовуватися м’які заходи (додаткова перевірка, журналювання, сповіщення аналітика). При високому рівні ризику система може ініціювати блокування сесії, примусовий вихід користувача, увімкнення багатофакторної автентифікації або інші захисні механізми.

Описана архітектура дозволяє розділити систему на незалежні модулі, кожен з яких може масштабуватися, модифікуватися або замінюватися без зміни загальної логіки роботи методу [25]. Модуль збору логів може бути інтегрований з існуючою системою журналювання вебзастосунку; модуль попередньої обробки - адаптований під формат конкретних логів; модуль формування ознак - доповнений новими показниками, релевантними для певного класу атак.

Водночас наявність окремого класифікаційного шару, який оперує інтегральним показником аномальності та вектором ознак, забезпечує гнучкість і можливість подальшого вдосконалення моделі без зміни нижчого рівня збирання даних. Така побудова архітектури створює основу для реалізації масштабованої та адаптивної системи оцінювання безпеки вебзастосунків, здатної враховувати як відомі, так і нові типи загроз.

3.2 Формування аналітичних таблиць для оцінювання поведінкових ризиків

У межах реалізації методу оцінювання безпеки вебзастосунків було створено окремий аналітичний рівень даних, що містить структуровані, агреговані та збагачені показники, необхідні для побудови поведінкових ознак та застосування моделей машинного навчання [17]. На цьому рівні дані подаються у вигляді логічно завершених таблиць, кожна з яких відображає певний аспект взаємодії користувачів і зовнішніх сутностей із вебзастосунком.

На відміну від сирих логів, аналітичні таблиці містять узагальнену інформацію, що значно полегшує процес обчислення інтегрального показника аномальності та формування RiskScore. Структура цих таблиць була спроєктована з урахуванням вимог до систем поведінкового аналізу та особливостей архітектури Databricks. Нижче наведено опис створених аналітичних таблиць, до яких додано SQL-визначення у вигляді відповідних рисунків.

```
4   CREATE OR REPLACE TABLE databricks_kanap_workspace.security.user_sessions_gold (
5       session_id STRING,
6       user_id STRING,
7       ip_address STRING,
8       user_agent STRING,
9       total_requests INT,
10      failed_auth_attempts INT,
11      unique_endpoints INT,
12      avg_request_length DOUBLE,
13      time_spent_seconds DOUBLE,
14      abnormal_activity_score DOUBLE,
15      risk_score DOUBLE,
16      event_created_at TIMESTAMP
17  )
18  USING DELTA;
```

Рис 3.2 Структура таблиці користувацьких сесій

Поля таблиці включають (див. рис. 3.2):

- загальна кількість запитів (total_requests);
- кількість невдалих авторизацій (failed_auth_attempts);
- кількість унікальних endpoint-ів (unique_endpoints);
- середня довжина запиту (avg_request_length);
- тривалість сесії (time_spent_seconds);
- інтегральний показник аномальності (abnormal_activity_score);
- оцінений ризик (risk_score).

```
22  CREATE OR REPLACE TABLE databricks_kanap_workspace.security.ip_activity_gold (
23      ip_address STRING,
24      geo_country STRING,
25      total_requests INT,
26      unique_users INT,
27      error_rate DOUBLE,
28      suspicious_requests INT,
29      blocked_requests INT,
30      request_frequency_per_min DOUBLE,
31      risk_score DOUBLE,
32      event_created_at TIMESTAMP
33  )
34  USING DELTA;
```

Рис 3.3 Структура таблиці активності IP-адрес

Поля таблиці включають (див. рис. 3.3):

- загальний та відносний обсяг трафіку (total_requests, request_frequency_per_min);
- поведінкові аномалії (error_rate, suspicious_requests);
- кількість унікальних користувачів, які працюють з IP (unique_users);
- фінальну оцінку ризику (risk_score).

```
38  CREATE OR REPLACE TABLE  databricks_kanap_workspace.security.user_behavior_gold (
39      user_id STRING,
40      avg_request_interval DOUBLE,
41      distinct_urls INT,
42      failed_logins INT,
43      successful_logins INT,
44      unusual_time_activity BOOLEAN,
45      endpoint_entropy DOUBLE,
46      avg_payload_size DOUBLE,
47      risk_score DOUBLE,
48      event_created_at TIMESTAMP
49  )
50  USING DELTA;
```

Рис 3.4 Структура таблиці поведінки користувачів

Поля таблиці включають (див. рис. 3.4):

- середній інтервал між запитами (avg_request_interval);

- ентропію маршрутів (endpoint_entropy);
- успішні та невдалі входи (successful_logins, failed_logins);
- активність у нетипові години (unusual_time_activity);
- середній розмір переданих даних (avg_payload_size).

```
54  CREATE OR REPLACE TABLE  databricks_kanap_workspace.security.auth_events_gold (
55      auth_event_id STRING,
56      user_id STRING,
57      session_id STRING,
58      ip_address STRING,
59      login_status STRING,
60      device_fingerprint STRING,
61      login_hour INT,
62      geo_country STRING,
63      risk_score DOUBLE,
64      event_created_at TIMESTAMP
65  )
66  USING DELTA;
```

Рис 3.5 Структура таблиці подій автентифікації

Поля таблиці включають (див. рис. 3.5):

- входи з нових пристроїв (device_fingerprint);
- підозрілі часові інтервали (login_hour);
- географічні аномалії (geo_country);
- статус успішності входу.

```
CREATE OR REPLACE TABLE  databricks_kanap_workspace.security.web_events_gold (
    event_id STRING,
    session_id STRING,
    user_id STRING,
    ip_address STRING,
    http_method STRING,
    url STRING,
    response_code INT,
    request_size_bytes INT,
    response_time_ms DOUBLE,
    contains_suspicious_pattern BOOLEAN,
    anomaly_flag BOOLEAN,
    risk_score DOUBLE,
    event_created_at TIMESTAMP
)
USING DELTA;
```

Рис 3.6 Структура таблиці вебподій

Поля таблиці включають (див. рис. 3.6):

- метод і URL запиту (http_method, url);
- код відповіді (response_code);
- параметри продуктивності (response_time_ms);
- сигнали низькорівневих аномалій (contains_suspicious_pattern, anomaly_flag);
- оцінений ризик (risk_score).

```
CREATE OR REPLACE TABLE databricks_kanap_workspace.security.ml_sessions_training (
  session_id STRING,
  user_id STRING,
  ip_address STRING,
  total_requests INT,
  failed_auth_attempts INT,
  unique_endpoints INT,
  avg_request_length DOUBLE,
  time_spent_seconds DOUBLE,
  abnormal_activity_score DOUBLE,
  ip_total_requests INT,
  ip_error_rate DOUBLE,
  user_failed_logins INT,
  user_successful_logins INT,
  unusual_time_activity BOOLEAN,
  label INT
)
USING DELTA;
```

Рис 3.7 Структура таблиці для тренування моделі

Поля таблиці включають (див. рис. 3.7):

- усі поведінкові ознаки;
- rule-based індикатори;
- лейбл (label) - мітку «нормальна / аномальна сесія».

Ця таблиця на рис 3.7 є основою для навчання Random Forest.

```
CREATE OR REPLACE TABLE databricks_kanap_workspace.security.session_risk_scores (
  session_id STRING,
  user_id STRING,
  ip_address STRING,
  risk_score DOUBLE,
  prediction INT,
  model_version STRING,
  scored_at TIMESTAMP,
  event_created_at TIMESTAMP
)
USING DELTA;
```

Рис 3.8 Структура таблиці результатів RiskScore

Поля таблиці включають (див. рис. 3.8):

- risk_score - ймовірність аномальної активності;
- prediction - рішення моделі (0/1);
- model_version - контроль і відтворюваність;
- час оцінювання (scored_at).

Цей набір даних використовується системою безпеки у Production.

Створені аналітичні таблиці відіграють ключову роль у роботі всієї системи, оскільки саме вони забезпечують формування поведінкових ознак високої якості. Завдяки детальній структурі та агрегуванню даних на різних рівнях - від окремих вебподій до узагальнених характеристик користувачів, сесій та IP-адрес -система отримує повноцінні та інформативні параметри, які відображають реальні патерни взаємодії з вебзастосунком. Це дозволяє не лише зафіксувати факт певної події, а й інтерпретувати її у ширшому контексті, визначивши, чи є вона типовою або такою, що відхиляється від нормальної поведінки [14].

Структура таблиць також створює умови для точного обчислення інтегрального показника AAS. Оскільки всі необхідні параметри зібрані в уніфікованому вигляді, модель отримує послідовний та чистий набір даних, що дозволяє зменшити похибки, пов’язані з неповнотою або неоднорідністю логів. Це особливо важливо для поведінкового аналізу, де навіть незначні зміни у значеннях ознак можуть суттєво вплинути на результат. Впорядкованість і узгодженість даних забезпечують стабільність формули AAS та її коректне застосування до широкого спектра користувацьких сценаріїв.

Крім того, створені таблиці забезпечують повноцінну підтримку моделі машинного навчання. Дані подані у форматі, що оптимізований для навчання класифікаторів, а тому модель отримує не сирі логи, а узагальнені вектори ознак, з яких усунуто шум, дублікати та технічні спотворення. Це підвищує точність прогнозування, спрощує процедуру навчання та створює передумови для використання різних типів алгоритмів, включно з ансамблевими методами, які потребують добре структурованих і статистично збалансованих входів. Ще однією важливою властивістю розроблених таблиць є забезпечення централізованого доступу до всіх ризикових метрик. Усі показники, які впливають на оцінювання безпеки - від активності IP-адрес до тривалості сесій і частки помилкових запитів - зібрані у зручному вигляді, що дозволяє оперативно виконувати моніторинг, порівняння й аналіз. Це підвищує керованість системи, полегшує розслідування інцидентів та створює єдине джерело правди (single source of truth) для всіх модулів.

Окремо варто підкреслити масштабованість та прозорість аналізу. Аналітичні таблиці побудовані таким чином, щоб легко адаптуватися до збільшення обсягів даних, появи нових типів ознак або розширення формату логів. Це дозволяє системі розвиватися разом із вебзастосунком і залишатися актуальною навіть у разі зміни архітектури чи додавання нових функцій. Прозора структура даних забезпечує простоту аудиту, можливість детального перегляду кожного етапу обробки та швидке виявлення потенційних проблем.

Таким чином, сформований аналітичний рівень даних виступає фундаментальним компонентом системи оцінювання безпеки вебзастосунків. Він забезпечує надійну основу для подальшої обробки даних, математичного моделювання, роботи алгоритмів машинного навчання та прийняття рішень щодо рівня ризику. Без цієї структурованої та узгодженої бази даних реалізація повноцінної системи поведінкового аналізу була б неможливою або суттєво менш ефективною.

3.3 Реалізація моделі машинного навчання

Реалізація моделі машинного навчання є ключовим етапом побудови системи оцінювання безпеки вебзастосунків, оскільки саме на цьому рівні сформовані поведінкові ознаки, інтегральний показник аномальності та додаткові rule-based сигнали перетворюються на конкретне числове рішення щодо рівня ризику кожної користувацької сесії. Проєктовану модель було реалізовано в середовищі Databricks із застосуванням бібліотеки PySpark ML, що дозволило масштабувати аналіз та обробку великих обсягів логів, забезпечити високу продуктивність обчислень та інтегрувати процес навчання у загальну архітектуру аналітичного конвеєра [13].

Першим кроком є формування тренувального набору даних, який міститься у спеціально створеній таблиці ml_sessions_training. Саме вона виступає джерелом структурованих і збалансованих даних, де кожен рядок відповідає окремій користувацькій сесії, описаній набором числових та логічних ознак. До складу цього набору входять параметри інтенсивності взаємодії, частотні характеристики IP-адреси, поведінкові особливості користувача, результати rule-based аналізу та інтегральний показник AAS, який узагальнює ключові поведінкові аномалії. Для кожної сесії також міститься мітка класу, що визначає, чи є вона нормальною або аномальною. Формування такої таблиці забезпечує якісну основу для supervised learning і дозволяє моделі навчатися на реальних прикладах поведінки.

Під час попередньої обробки даних здійснюється вибір релевантних ознак, нормалізація числових параметрів і приведення категоріальних полів до формату, придатного для машинного навчання. Застосування VectorAssembler дає змогу об’єднати всі значення ознак у вектор, який використовується класифікатором Random Forest. Така підготовка забезпечує узгодженість структури даних і дозволяє уникнути зміщення моделі, спричиненого відмінностями між масштабами ознак.

Вибір моделі Random Forest зумовлений низкою переваг цього алгоритму. По-перше, він добре працює з великою кількістю вхідних ознак, що особливо важливо для поведінкового аналізу, де кожна сесія характеризується десятками параметрів. По-друге, Random Forest стійкий до шуму та пропусків у даних завдяки ансамблевій природі: рішення приймається на основі сукупності дерев, що знижує ризик переобучення. По-третє, модель дозволяє оцінювати важливість окремих ознак, що є корисним у контексті кібербезпеки, де важливо розуміти, які саме поведінкові фактори найбільше впливають на прогноз.

Після навчання моделі здійснюється оцінювання її ефективності на тестовій підвибірці [23]. Основними метриками є точність, F1-міра, precision та recall, але особлива увага приділяється показнику recall для аномального класу. У системах безпеки пропуск шкідливої активності є критичною помилкою, тому модель налаштовується таким чином, щоб мінімізувати ризик хибно-негативних результатів. Результати тестування показали, що побудована модель демонструє високу здатність виявляти аномальні патерни поведінки користувачів, зберігаючи при цьому прийнятний рівень хибних спрацювань.

Після вибору оптимальних гіперпараметрів модель реєструється в MLflow Model Registry, що забезпечує відтворюваність та контроль версій. Реєстрація моделі дозволяє легко інтегрувати її в інші компоненти системи, виконувати оновлення, порівнювати результати різних експериментів та слідкувати за якістю моделі в довгостроковій перспективі. Завдяки можливостям Databricks створюється повний цикл MLOps - від формування даних до застосування моделі в робочому середовищі.

У фазі експлуатації модель використовується для обчислення RiskScore у режимі пакетної або потокової обробки. Кожна нова користувацька сесія проходить ті самі етапи, що й дані в тренувальному наборі: формування поведінкових ознак, обчислення AAS, створення вектора параметрів та прогнозування рівня ризику. Результати класифікації зберігаються у спеціальній таблиці, що дозволяє здійснювати моніторинг ризикових подій, повторний аналіз, аудит та подальший розвиток системи. Таким чином, реалізація моделі машинного навчання поєднує поведінковий аналіз, rule-based логіку та статистичні методи, створюючи адаптивний механізм виявлення аномалій, який здатний ефективно реагувати на складні сценарії атак і забезпечувати високий рівень безпеки вебзастосунку.

3.4 Оцінка отриманих результатів та ефективності методу

У даному підпункті виконується узагальнення експериментальних результатів та аналіз ефективності запропонованого методу оцінювання безпеки вебзастосунків. Проведене дослідження мало на меті визначити, наскільки поведінковий аналіз у поєднанні з моделями машинного навчання здатний покращити виявлення аномальної активності порівняно з традиційними методами контролю безпеки. Для цього було сформовано та протестовано декілька груп підходів: rule-based механізми, методи статичного й динамічного аналізу (SAST/DAST), механізми лімітування IP-адрес та побудовану модель Random Forest. Усі підходи було оцінено у межах єдиної тестової вибірки, що дозволило отримати об’єктивне порівняння. На рисунку 3.12 наведено результати порівняння точності кожного із розглянутих методів.

Розглядаючи дані, можна відзначити, що традиційні rule-based системи демонструють обмежену результативність із показником точності близько 0.61. Це пояснюється тим, що такі системи реагують виключно на заздалегідь прописані шаблони поведінки та сигнатури, а отже, мають низьку здатність до узагальнення та фактично не здатні виявляти нові або модифіковані типи атак. Вони

залишаються ефективними лише у стабільних, добре передбачуваних сценаріях, де ризики вже відомі та задокументовані.

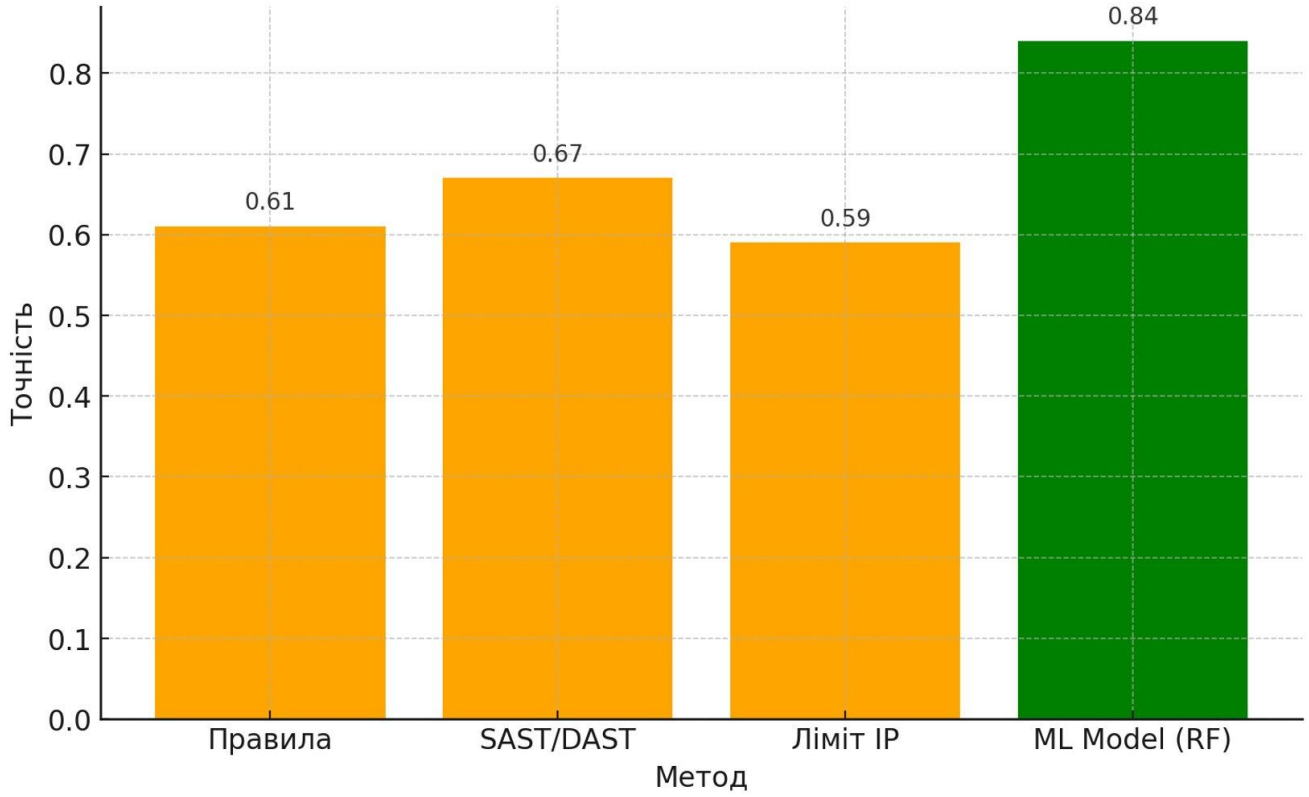


Рис 3.9 Порівняння точності методів оцінювання безпеки

Методи статичного (SAST) та динамічного (DAST) аналізу виявилися дещо результативнішими, досягнувши точності на рівні 0.67. Це очікувано, оскільки такі інструменти допомагають виявляти складні вразливості у коді або поведінці програми під час виконання. Проте їхній підхід орієнтовано на аналіз самої системи, а не на поведінку користувачів, тому вони залишаються непридатними для виявлення аномальних сценаріїв, що виникають у реальному середовищі при взаємодії з вебзастосунком.

Методи лімітування активності на рівні IP-адреси показали точність на рівні 0.59. Це найнижчий показник серед розглянутих підходів. Основна причина полягає в тому, що сучасні зловмисники активно використовують проксі, VPN, розподілені мережі та динамічні IP-пули. У результаті блокування або обмеження активності лише на основі IP не є ефективним: система або блокує легітимних користувачів (фальшиві спрацювання), або не виявляє реальних атак (пропуск

загроз). Побудована модель машинного навчання Random Forest показала значно вищий рівень точності - 0.84. Таке зростання ефективності досягається завдяки використанню великого набору поведінкових ознак, що включає характеристики сесій, частоти запитів, помилкові спроби автентифікації, аномалії у параметрах запитів, інтегральний показник AAS та інші метрики. Модель здатна розпізнавати складні зв'язки між ознаками, враховувати часову динаміку та визначати відхилення, які неможливо описати набором жорстких правил.

Random Forest також має високу стійкість до шуму та є інтерпретованим у контексті важливості ознак, що є важливим фактором у системах кіберзахисту. Завдяки ансамблевій природі модель розглядає десятки різних «рішень» (дерев), що дозволяє мінімізувати ймовірність хибних спрацювань та водночас ефективно виявляти аномалії, які повторюються у кількох незалежних ознаках.

Отримані результати підтверджують, що запропонований метод значно перевершує традиційні підходи за двома ключовими критеріями:

1. Здатність виявляти нові та нетипові атаки. Модель навчається на реальних даних і не обмежується фіксованими сигнатурами, що дозволяє їй адаптуватися до змін у поведінці користувачів та зловмисників.
2. Комплексний характер аналізу. Метод враховує характеристики не лише окремих запитів, а й повну поведінку користувацької сесії, історію подій, параметри IP-адреси та rule-based сигнали. Таким чином, модель працює не з ізольованими фактами, а з контекстом.

Підсумовуючи проведений аналіз, можна дійти висновку, що запропонований метод оцінювання безпеки забезпечує значне підвищення точності виявлення аномальної активності та є придатним для застосування у сучасних вебсистемах. Його використання дозволяє істотно зменшити кількість пропущених атак, забезпечити адаптивність до нових загроз та знизити навантаження на rule-based системи завдяки автоматизованому поведінковому аналізу. Результати експериментів демонструють, що інтеграція машинного навчання у процес оцінювання безпеки є ефективним напрямом розвитку захисних механізмів вебзастосунків.

ВИСНОВКИ

У ході виконання дипломної роботи було здійснено дослідження сучасних загроз вебзастосунків, обмежень традиційних методів оцінювання безпеки та можливостей використання машинного навчання для виявлення аномальної активності. Аналіз існуючих підходів показав, що rule-based системи, сигнатурні механізми та інструменти статичного й динамічного аналізу залишаються важливими складниками кіберзахисту, проте не забезпечують належного рівня адаптивності в умовах, коли велика частина атак не має чіткої сигнатури або маскується під легітимну взаємодію. Це підтвердило актуальність дослідження та необхідність розробки інтелектуальних методів, що враховують поведінкові особливості користувачів і мережевих подій.

У межах роботи було сформовано набір поведінкових ознак для оцінювання сесій користувачів, побудовано математичну модель інтегрального показника аномальності (AAS) та визначено ймовірнісний показник ризику RiskScore. Запропонована модель поєднує rule-based індикатори, статистичні характеристики поведінки та ансамблевий алгоритм Random Forest, що дозволяє враховувати складні залежності між ознаками й приймати рішення на підставі сукупного аналізу подій. Розроблена архітектура передбачає створення аналітичних таблиць у середовищі Databricks, підготовку датасету для моделі та автоматизоване формування ризикової оцінки для кожної користувацької сесії.

Експериментальні дослідження продемонстрували суттєву перевагу запропонованого методу над традиційними підходами. Rule-based механізми показали точність на рівні 0.61, підходи SAST/DAST - 0.67, IP-лімітування - 0.59. У той час модель Random Forest досягла точності 0.84, що свідчить про її здатність ефективно розпізнавати як очевидні, так і приховані поведінкові аномалії. Перевірка результатів підтвердила, що інтеграція поведінкових ознак, rule-based сигналів і машинного навчання дає змогу значно знизити кількість пропущених

атак та зменшити рівень хибних спрацювань. Це робить запропонований підхід більш адаптивним, гнучким і точним порівняно з класичними методами контролю безпеки.

Отримані результати демонструють, що розроблений метод оцінювання безпеки може бути використаний як основа для побудови автоматизованих систем моніторингу вебзастосунків.

Завдяки можливості подальшого донавчання моделі, розширення набору ознак і інтеграції з потоковими платформами система може працювати в умовах реального часу та адаптуватися до нових сценаріїв атак. Метод є масштабованим і може бути інтегрованим в архітектури сучасних вебсистем, що підкреслює його практичну цінність.

Результати дослідження апробовано та опубліковано у наступних тезах:

1. Конопластий А.Р., Золотухіна О.А. Дослідження моделей та програмних компонентів системи оцінки безпеки веб-застосувань. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 424.
2. Конопластий А.Р., Золотухіна О.А. Використання моделей машинного навчання для автоматизованої оцінки безпеки веб-застосувань. V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2025. С. 27.

ПЕРЕЛІК ПОСИЛАНЬ

1. Shahid, M. Machine Learning for Detection and Mitigation of Web Vulnerabilities and Web Attacks. arXiv preprint, 2023. [Електронний ресурс]. URL: https://arxiv.org/abs/2304.14451
2. Farasat, T., Ahmadzai, A., George, A. E., Qaderi, S. A., Dordevic, D., & Posegga, J. SafePyScript: A Web-Based Solution for Machine Learning-Driven Vulnerability Detection in Python. arXiv preprint, 2024. [Електронний ресурс]. URL: https://arxiv.org/abs/2411.00636
3. Kshetri, N., Kumar, D., Hutson, J., Kaur, N., & Osama, O. F. algoXSSF: Detection and analysis of cross-site request forgery (XSRF) and cross-site scripting (XSS) attacks via Machine learning algorithms. arXiv preprint, 2024. [Електронний ресурс]. URL: https://arxiv.org/abs/2402.01012
4. Kumar, P., Antony, K., Banga, D., & Sohal, A. PhishNet: A Phishing Website Detection Tool using XGBoost. arXiv preprint, 2024. [Електронний ресурс]. URL: https://arxiv.org/abs/2407.04732
5. Wen, Y. A Metamodel for Web Application Security Evaluation. FRUCT, 2023. [Електронний ресурс]. URL: https://fruct.org/publications/volume-34/fruct34/files/Wen.pdf
6. OWASP Foundation. OWASP Machine Learning Security Top Ten. OWASP, 2023. [Електронний ресурс]. URL: https://owasp.org/www-project-machine-learning-security-top-10/
7. OWASP Foundation. OWASP Application Security Verification Standard (ASVS). OWASP, 2023. [Електронний ресурс]. URL: https://owasp.org/www-project-application-security-verification-standard/
8. OWASP Foundation. OWASP AI Security and Privacy Guide. OWASP, 2023. [Електронний ресурс]. URL: https://owasp.org/www-project-ai-security-and-privacy-guide/
9. Cisco Systems. What Is Machine Learning in Security? Cisco, 2025. [Електронний ресурс]. URL:

https://www.cisco.com/c/en/us/products/security/machine-learning-security.html

10. Herrin, C. Cybersecurity execs face a new battlefront: 'It takes a good-guy AI to fight a bad-guy AI'. Business Insider, 2025. [Електронний ресурс]. URL: https://www.businessinsider.com/artificial-intelligence-cybersecurity-large-language-model-threats-solutions-2025-5
11. Software Testing Magazine. Next-Gen Security Testing: How AI and Machine Learning Are Shaping Cybersecurity. Software Testing Magazine, 2025. [Електронний ресурс]. URL: https://www.softwaretestingmagazine.com/knowledge/next-gen-security-testing-how-ai-and-machine-learning-are-shaping-cybersecurity/
12. Barracuda Networks. 5 ways AI is being used to improve security: Application security. Barracuda Blog, 2024. [Електронний ресурс]. URL: https://blog.barracuda.com/2024/07/12/5-ways-ai-is-being-used-to-improve-security--application-securit
13. WebAsha Technologies. Using AI for Vulnerability Assessments in Web Applications. WebAsha Blog, 2025. [Електронний ресурс]. URL: https://www.webasha.com/blog/using-ai-for-vulnerability-assessments-in-web-applications-enhancing-security-with-smart-automation
14. Bishop Fox. AI/ML and LLM Security Assessments. Bishop Fox, 2024. [Електронний ресурс]. URL: https://bishopfox.com/services/ai-ml-security-assessment
15. Akamai Technologies. Harnessing Artificial Intelligence for a Superior Web Application Firewall. Akamai Blog, 2025. [Електронний ресурс]. URL: https://www.akamai.com/blog/security/harnessing-artificial-intelligence-for-superior-web-application-firewall
16. Infosys. How AI is enhancing web application security. Infosys Blogs, 2025. [Електронний ресурс]. URL: https://blogs.infosys.com/digital-experience/web-ui-ux/how-ai-is-enhancing-web-application-security.html

17. Rapid7. AI and ML Can Be Used to Stop DAST Attacks Before They Start. Rapid7 Blog, 2023. [Електронний ресурс]. URL: https://www.rapid7.com/blog/post/2023/11/09/artificial-intelligence-and-machine-learning-can-be-used-to-stop-dast-attacks-before-they-start/
18. ACCELQ. AI Application Security Testing for Smarter Approach. ACCELQ Blog, 2024. [Електронний ресурс]. URL: https://www.accelq.com/blog/ai-application-security-testing/
19. Kiuwan. How AI Is Changing Application Security Testing. Kiuwan Blog, 2023. [Електронний ресурс]. URL: https://www.kiuwan.com/blog/ai-how-it-is-changing-application-security-testing/
20. The Smart Scanner. How AI and Machine Learning are Revolutionizing Web Security. The Smart Scanner Blog, 2023. [Електронний ресурс]. URL: https://www.thesmartscanner.com/blog/how-ai-and-machine-learning-are-revolutionizing-web-security
21. ZeroThreat AI. AI in AppSec: Enhancing Cybersecurity & Threat Detection. ZeroThreat AI Blog, 2025. [Електронний ресурс]. URL: https://zerothreat.ai/blog/ai-is-revolutionizing-application-security
22. LinkedIn. How AI and Machine Learning Are Transforming Web Application Security. LinkedIn Pulse, 2025. [Електронний ресурс]. URL: https://www.linkedin.com/pulse/how-ai-machine-learning-transforming-web-application-lqpxf
23. Schutzwerk. Web Application Security Assessment. Schutzwerk, 2025. [Електронний ресурс]. URL: https://www.schutzwerk.com/en/assessment/web-application-security-assessment/
24. Black Duck Software. What Is Web Application Security and How Does It Work? Black Duck, 2025. [Електронний ресурс]. URL: https://www.blackduck.com/glossary/what-is-web-application-security.html
25. NIST. Secure Software Development Framework (SSDF), SP 800-218. National Institute of Standards and Technology, 2023. [Електронний ресурс]. URL: https://csrc.nist.gov/publications/detail/sp/800-218/final

26. OWASP Developer Guide (PDF). OWASP (Open Web Application Security Project) Official Documentation. [Електронний ресурс]. URL: https://owasp.org/www-project-developer-guide/assets/exports/OWASP_Developer_Guide.pdf
27. Wiz.io. Application Security Frameworks and Standards: OWASP, NIST, ISO. Wiz Academy, 2025. [Електронний ресурс]. URL: https://www.wiz.io/academy/application-security-frameworks
28. Sarrah Pitaliya. Decoding The Future of Web App and API Security Testing. Medium, 2024. [Електронний ресурс]. URL: https://sarrahpitaliya.medium.com/decoding-the-future-of-web-app-and-api-security-testing-6c40836f1ddc
29. University of Texas Libraries. Using machine learning to detect web application attacks. University of Texas, 2025. [Електронний ресурс]. URL: https://repositories.lib.utexas.edu/items/e5d48b6b-675f-4c20-aa34-ded320122a9e
30. Business Insider. Cybersecurity execs face a new battlefront: 'It takes a good-guy AI to fight a bad-guy AI'. Business Insider, 2025. [Електронний ресурс]. URL: https://www.businessinsider.com/artificial-intelligence-cybersecurity-large-language-model-threats-solutions-2025-5
31. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. MIT Press, 2016. [Електронний ресурс]. URL: https://www.deeplearningbook.org/

ДОДАТОК А. ДЕМОНСТРАТИВНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«Методика оцінювання безпеки вебзастосунків із використанням машинного навчання»

Виконав: студент групи ПДМ-61 Андрій КОНОПЛЯСТИЙ

Керівник: канд. техн. наук, доцент кафедри ІПЗ Оксана ЗОЛОТУХІНА

Київ - 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення ефективності оцінювання безпеки вебзастосунків на основі використання алгоритмів машинного навчання для виявлення, аналізу та класифікації аномальної активності.

Об'єкт дослідження: процес оцінювання безпеки вебзастосунків

Предмет дослідження: методика оцінювання безпеки вебзастосунків на основі використання алгоритмів машинного навчання для виявлення, аналізу та класифікації аномальної активності.

2

АКТУАЛЬНІСТЬ РОБОТИ

Метод / Підхід	Суть підходу	Проблема в контексті безпеки вебзастосунків
Rule-based (порогові правила)	Реагує на перевищення встановлених лімітів: частоти помилок, кількості запитів, часу між запитами.	Не розуміють поведінку користувача, легко обходяться, часто дають хибні спрацювання.
Сигнатурні методи (SAST/DAST)	Знаходять відомі вразливості у коді або під час тестування роботи вебзастосунку.	Не виявляють нові атаки, не працюють у реальному часі, не враховують поведінку користувача.
IP-rate limiting / Firewall	Контролює трафік за IP-адресою, обмежує або блокує підозрілу активність.	Не бачать контексту сесії, не відрізняють легітимних користувачів, слабкі проти бот-мереж.
Моніторинг та лог-аналіз	Виявляє певні події або аномалії за заздалегідь заданими правилами.	Потребують ручної настройки та часто створюють багато хибних сповіщень.
DevSecOps-практики	Інтегрують безпеку у процес розробки та тестування застосунку.	Орієнтовані на код, а не на поведінку користувача

3

ЗАВДАННЯ РОБОТИ

- Проаналізувати існуючі підходи до оцінювання безпеки вебзастосунків і визначити їх ключові обмеження.
- Сформувати математичну модель, що описує оцінювання ризику користувацької сесії на основі поведінкових ознак.
- Розробити алгоритм визначення показника ризику, включаючи обчислення інтегрального індикатора аномальності та формування класової мітки.
- Побудувати модель машинного навчання, здатну автоматично класифікувати ризикові та нормальні сесії.
- Провести моделювання та порівняльний аналіз, щоб оцінити ефективність запропонованої методики у порівнянні з існуючими підходами.

4

МАТЕМАТИЧНА МОДЕЛЬ ОЦІНЮВАННЯ РИЗИКУ КОРИСТУВАЦЬКИХ СЕСІЙ

1.Базова формула оцінювання аномальності

Для кількісної оцінки поведінки користувача вводиться інтегральний показник **AAS (Anomalous Activity Score)**, який розраховується за формулою:

$$AAS = 0.25 \cdot min(\frac{FA}{10}, 1) + 0.20 \cdot min(ER, 1) + 0.20 \cdot min(\frac{UFL}{10}, 1) + 0.20 \cdot I(SP) + 0.15 \cdot I(A)$$

де

FA - кількість невдалих спроб входу;
ER - частка помилкових запитів, що надходять з IP-адреси;
UFL - невдалі логіни користувача;
SP - індикатор наявності підозрілих патернів у подіях;
A - виявлені аномалії у веб-логах;
I(x) - одинична функція, що приймає значення 1 у разі наявності події.

2.Правило формування класової мітки

На основі отриманого показника **AAS** визначається приналежність сесії до нормальної чи ризикової:

$$label = \begin{cases} 1, & AAS \geq 0.6 \text{ або } SP = 1 \text{ або } A = 1 \\ 0, & \text{в інших випадках} \end{cases}$$

5

МАТЕМАТИЧНА МОДЕЛЬ ОЦІНЮВАННЯ РИЗИКУ КОРИСТУВАЦЬКИХ СЕСІЙ (ПРОДОВЖЕННЯ)

3.Вектор ознак моделі

Для класифікації користувацьких сесій використовується ансамблева модель Random Forest, що працює з багатовимірним вектором параметрів:

$$X = [TR, FA, UE, AR, TS, AAS, ITR, IER, UFL, USL]$$

де:

TR - загальна кількість HTTP-запитів у сесії;
FA - невдалі спроби автентифікації;
UE - кількість унікальних endpoint-ів;
AR - середня довжина запиту;
TS - тривалість активності;
AAS - інтегральний показник аномальності (формула (1));
ITR - кількість запитів від IP;
IER - частка помилкових запитів з IP;
UFL - невдалі входи користувача;
USL - успішні входи користувача.

4.Формування підсумкового показника ризику

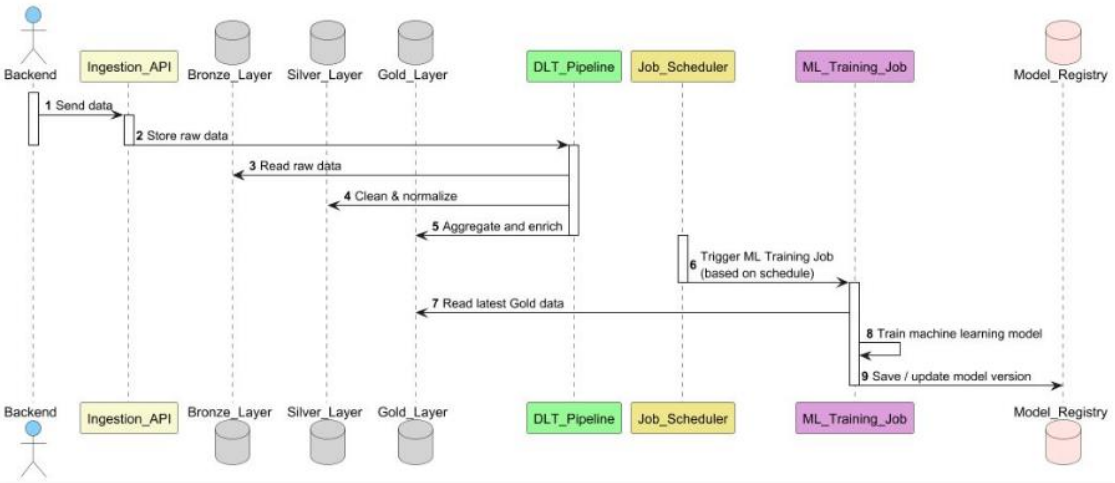
Модель Random Forest повертає ймовірність належності сесії до класу «ризикова»:

$$RiskScore = P \cdot (label = 1 \mid X)$$

Отриманий *RiskScore* використовується як основний критерій для автоматичного виявлення небезпечних користувацьких сесій.

6

ПРАКТИЧНИЙ РЕЗУЛЬТАТ



Діаграма послідовності взаємодії компонентів під час обчислення показника ризику (RiskScore)

7

ПРАКТИЧНИЙ РЕЗУЛЬТАТ

```
163  select * from databricks_kanap_workspace.security.session_risk_scores order by risk_score desc;
164
165
```

Add parameter

Results 9 of 9 | Table | +

	session_id	user_id	ip_address	1.2 risk_score	prediction	model_version
1	b9e0ad12-51f4-4cd1-b223-4ccfa241e011	f8ab2cbe-1e8f-4fc2-af91-d1ac901a0011	45.87.122.12	0.95	2	v1.1
2	4bd1c7a4-5e77-49b9-9320-7bcfa241e005	f2a50c5f-7b3c-4dc1-af31-d1ac901a0005	172.16.0.21	0.91	2	v1.0
3	33da9ec1-7a2e-4f19-8c21-8acfa241e015	d2ef6a0e-7c3f-4dc2-afd1-d1ac901a0015	91.200.12.21	0.88	2	v1.2
4	7f4b19aa-9d22-4e3b-9011-2acfa241e003	d9f30b3f-9d2a-4cce-af10-d1ac901a0003	10.0.0.5	0.82	2	v1.0
5	6be3df82-1e2b-45f1-9322-2bcfa241e018	a5b29d3e-4f6f-4ac2-bf01-d1ac901a0018	46.118.55.44	0.79	2	v1.2
6	83ae21b4-8f33-4ef1-8b22-4dcfa241e020	c7d4bf5e-2b8f-4cc2-bf21-d1ac901a0020	92.60.181.11	0.77	2	v1.3
7	a8d19cfe-4cc2-45d0-9a22-0ccfa241e008	c5d80f8e-4d5e-4bc1-af61-d1ac901a0008	83.221.44.13	0.73	2	v1.1
8	a5c043d6-af56-4df2-8d44-6fcfa241e022	e9f6e17e-0d1f-4ec2-bf41-d1ac901a0022	102.44.19.89	0.68	2	v1.3

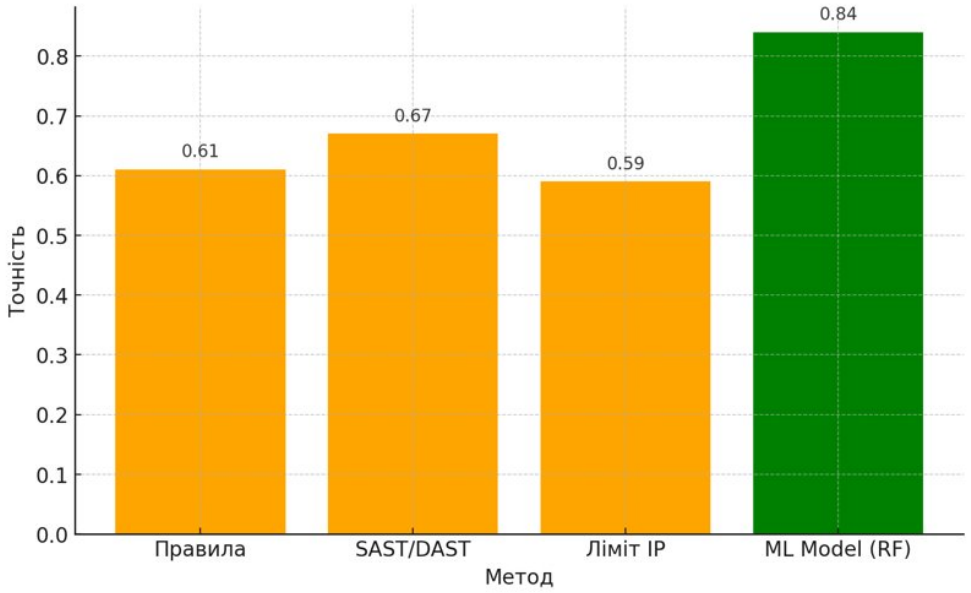
323 rows | 45.96s runtime

Refreshed 1 minute a

Результати користувацьких сесій за показником RiskScore

8

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ МОДЕЛІ



9

ВИСНОВКИ

1. На основі аналізу існуючих підходів обґрунтовано необхідність використання поведінкових ознак та методів машинного навчання для оцінювання безпеки вебзастосунків.
2. Розроблено математичну модель, яка формалізує ризик користувацької сесії через інтегральний показник аномальності.
3. Розроблено алгоритм визначення рівня ризику сесії, що передбачає обчислення показника аномальності та автоматичне формування навчальної вибірки без ручної розмітки.
4. Побудована модель Random Forest, яка забезпечила високу точність класифікації ризикових сесій та суттєво перевищила традиційні rule-based і сигнатурні методи.
5. Результати моделювання продемонстрували ефективність запропонованої методики та її здатність підвищувати рівень безпеки вебзастосунків. Порівняльний аналіз підтвердив перевагу використання машинного навчання над статичними та сигнатурними підходами у задачах виявлення аномальної поведінки.

10

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Коноплястий А.Р., Золотухіна О.А. Дослідження моделей та програмних компонентів системи оцінки безпеки веб-застосувань. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 424.

2. Коноплястий А.Р., Золотухіна О.А. Використання моделей машинного навчання для автоматизованої оцінки безпеки веб-застосувань. V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 27.

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
# Databricks notebook source

import mlflow
import mlflow.spark
from pyspark.sql import SparkSession, DataFrame, functions as F
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler, StandardScaler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.functions import vector_to_array

from pyspark.dbutils import DBUtils

from typing import Optional

CATALOG = "hive_metastore"
SCHEMA  = "security"
TRAIN_TBL = "ml_sessions_training"
SCORE_TBL = "session_risk_scores"
MODEL_NAME = "session_risk_model_rf"
LABEL_THRESHOLD = 0.6
EXPERIMENT = "/Shared/security_ml_experiment"

spark = SparkSession.builder.getOrCreate()
dbutils = DBUtils(spark)

class SecurityRiskJob:
   def __init__(self, spark: SparkSession):
       self.spark = spark
       self.features = [
           "total_requests", "failed_auth_attempts",
"unique_endpoints",
           "avg_request_length", "time_spent_seconds",
"abnormal_activity_score",
           "ip_total_requests", "ip_error_rate",
"user_failed_logins", "user_successful_logins"
       ]


   def get_task_parameter(self, key: str) -> Optional[str]:
       try:
           return dbutils.widgets.get(key)
       except Exception:
           return None

   def ensure_schema_and_tables(self) -> None:
       self.spark.sql(f"USE CATALOG {CATALOG}")
       self.spark.sql(f"CREATE SCHEMA IF NOT EXISTS {SCHEMA}")
       self.spark.sql(f"USE SCHEMA {SCHEMA}")
```

```
        # таблиці вже створюються окремо SQL-скриптом; залишимо
перевірку існування
        self.spark.sql(f"SHOW TABLES IN {SCHEMA}")

    def build_training_dataset(self) -> DataFrame:
        """
        Збирає train dataset з gold-рівня (тут очікуються таблиці:
        user_sessions_gold, ip_activity_gold, user_behavior_gold,
web_events_gold).
        Якщо їх назви інші — підправ у join-ах.
        """
        s = self.spark
        s.sql(f"USE CATALOG {CATALOG}")
        s.sql(f"USE SCHEMA {SCHEMA}")

        sessions = s.table("user_sessions_gold")
        ip = s.table("ip_activity_gold")
        users = s.table("user_behavior_gold")
        web = s.table("web_events_gold")

        anomalies = (
            web.groupBy("session_id")
               .agg(
                   F.max(F.col("anomaly_flag").cast("int")).alias("ha
s_anomaly"),
                   F.max(F.col("contains_suspicious_pattern").cast("i
nt")).alias("has_suspicious_pattern")
               )
        )

        base = (
            sessions.alias("s")
            .join(anomalies.alias("a"), "session_id", "left")
            .join(ip.alias("ip"), "ip_address", "left")
            .join(users.alias("u"), "user_id", "left")
        ).fillna({
            "failed_auth_attempts": 0,
            "ip_total_requests": 0,
            "ip_error_rate": 0.0,
            "user_failed_logins": 0,
            "user_successful_logins": 0,
            "has_anomaly": 0,
            "has_suspicious_pattern": 0
        })

        df = base.withColumn(
            "abnormal_activity_score",
            0.25 * F.least(F.col("failed_auth_attempts")/10.0,
F.lit(1.0)) +
            0.20 * F.least(F.col("ip_error_rate"), F.lit(1.0)) +
            0.20 * F.least(F.col("user_failed_logins")/10.0,
F.lit(1.0)) +
```

```
            0.20 * F.when(F.col("has_suspicious_pattern")==1,
1.0).otherwise(0.0) +
            0.15 * F.when(F.col("has_anomaly")==1,
1.0).otherwise(0.0)
        )

        df = df.withColumn(
            "label",
            F.when(
                (F.col("abnormal_activity_score") >=
F.lit(LABEL_THRESHOLD)) |
                (F.col("has_anomaly") == 1) |
(F.col("has_suspicious_pattern") == 1),
                1
            ).otherwise(0)
        )

        train_df = df.select(
            "session_id","user_id","ip_address",
            *self.features,
            "unusual_time_activity","label"
        )

        train_df.write.mode("overwrite").format("delta").saveAsTable(
TRAIN_TBL)
        return train_df

    def train_model(self, train_df: DataFrame):
        df = train_df.select(self.features + ["label"]).dropna()
        train, test = df.randomSplit([0.8, 0.2], seed=42)

        assembler = VectorAssembler(inputCols=self.features,
outputCol="features_raw")
        scaler = StandardScaler(inputCol="features_raw",
outputCol="features", withMean=True, withStd=True)
        rf = RandomForestClassifier(featuresCol="features",
labelCol="label", numTrees=120, maxDepth=10, seed=42)
        pipeline = Pipeline(stages=[assembler, scaler, rf])

        mlflow.set_experiment(EXPERIMENT)
        with mlflow.start_run(run_name="rf_training_job"):
            model = pipeline.fit(train)
            preds = model.transform(test)
            evaluator =
BinaryClassificationEvaluator(labelCol="label",
rawPredictionCol="rawPrediction")
            auc = evaluator.evaluate(preds)
            mlflow.log_metric("auc", auc)
            mlflow.spark.log_model(
                model,
                artifact_path="rf_model",
                registered_model_name=MODEL_NAME
```

```
            )
        return model

    def score_new_data(self) -> DataFrame:
        """
        Тягне останні дані, збирає фічі як у тренуванні, вантажить
модель із Registry,
        рахує risk_score та повертає DataFrame для запису.
        """
        s = self.spark
        s.sql(f"USE CATALOG {CATALOG}")
        s.sql(f"USE SCHEMA {SCHEMA}")

        sessions = s.table("user_sessions_gold")
        ip = s.table("ip_activity_gold")
        users = s.table("user_behavior_gold")
        web = s.table("web_events_gold")

        anomalies = (
            web.groupBy("session_id")
               .agg(
                   F.max(F.col("anomaly_flag").cast("int")).alias("ha
s_anomaly"),
                   F.max(F.col("contains_suspicious_pattern").cast("i
nt")).alias("has_suspicious_pattern")
               )
        )

        base = (
            sessions.alias("s")
            .join(anomalies.alias("a"), "session_id", "left")
            .join(ip.alias("ip"), "ip_address", "left")
            .join(users.alias("u"), "user_id", "left")
        ).fillna({
            "failed_auth_attempts": 0,
            "ip_total_requests": 0,
            "ip_error_rate": 0.0,
            "user_failed_logins": 0,
            "user_successful_logins": 0,
            "has_anomaly": 0,
            "has_suspicious_pattern": 0
        })

        feat = base.withColumn(
            "abnormal_activity_score",
            0.25 * F.least(F.col("failed_auth_attempts")/10.0,
F.lit(1.0)) +
            0.20 * F.least(F.col("ip_error_rate"), F.lit(1.0)) +
            0.20 * F.least(F.col("user_failed_logins")/10.0,
F.lit(1.0)) +
            0.20 * F.when(F.col("has_suspicious_pattern")==1,
1.0).otherwise(0.0) +
```

```
            0.15 * F.when(F.col("has_anomaly")==1,
1.0).otherwise(0.0)
        )

        feat = feat.select("session_id","user_id","ip_address",
*self.features)
        model_uri = f"models:/{MODEL_NAME}/Production"
        model = mlflow.spark.load_model(model_uri)

        scored = model.transform(feat)
        scored = (
            scored
            .withColumn("probability_arr",
vector_to_array(F.col("probability")))
            .withColumn("risk_score", F.col("probability_arr")[1])  #
ймовірність класу 1
            .withColumn("scored_at", F.current_timestamp())
            .withColumn("event_created_at", F.current_timestamp())
        )

        result = scored.select(
            "session_id", "user_id", "ip_address",
            "risk_score", "prediction",
            "probability_arr", "scored_at", "event_created_at"
        ).withColumnRenamed("probability_arr", "probability")

        return result

    def write_results(self, result_df: DataFrame) -> None:
        result_df.write.mode("append").format("delta").saveAsTable(SC
ORE_TBL)

    def run(self) -> None:
        self.ensure_schema_and_tables()
        train_df = self.build_training_dataset()
        self.train_model(train_df)
        scored = self.score_new_data()
        self.write_results(scored)

if __name__ == "__main__":
    job = SecurityRiskJob(spark)
    job.run()
```